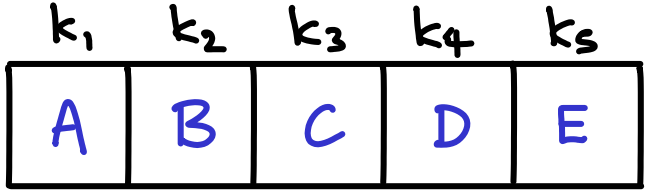
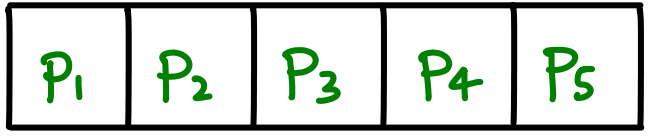
OPTIMAL BST : minimize expected search time

sorted keys:

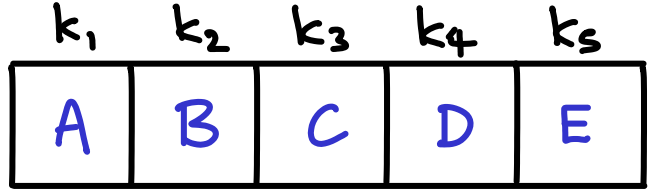


probability of searching each key:

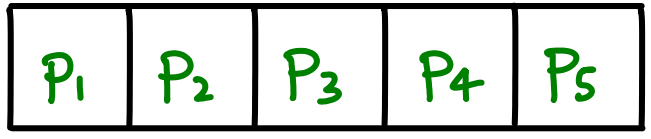


OPTIMAL BST : minimize expected search time

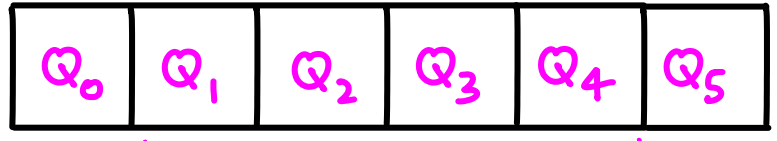
sorted keys:



probability of searching each key:



probability of searching value ≠ key:



too small between keys too big

OPTIMAL BST : minimize expected search time

sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

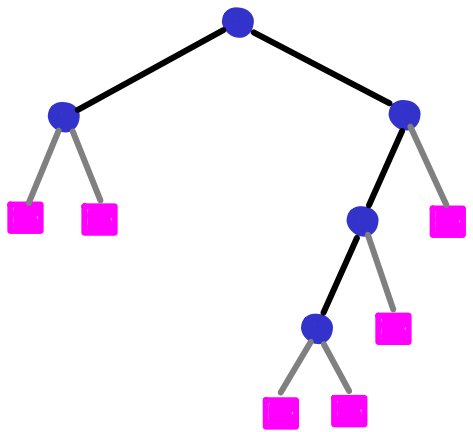
probability of searching each key:

p_1	p_2	p_3	p_4	p_5

probability of searching value $\neq$ key:

q_0	q_1	q_2	q_3	q_4	q_5

too small — between keys — too big



What is the best BST? (static)

→ "gaps" between keys must be leaves

OPTIMAL BST : minimize expected search time

sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

$\sum = 1$ {

search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key :

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

example from CLRS

OPTIMAL BST : minimize expected search time

E[time] = ?

sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

$\sum = 1$ {

search probability:

P_1	P_2	P_3	P_4	P_5
.15	.10	.05	.10	.20

probability ≠ key :

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
.05	.10	.05	.05	.05	.10

OPTIMAL BST : minimize expected search time

$E[\text{time}] =$

$\sum \text{prob}(i) \cdot \text{time}(i)$

p_i or q_i — depth(node i) (let root depth = 1)

$\sum = 1$ { search probability: ; probability ≠ key : }

sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

OPTIMAL BST : minimize expected search time

$E[\text{time}] =$

$$\sum \text{prob}(i) \cdot \text{time}(i)$$

$\text{prob}(i)$: p_i or q_i

$\text{time}(i)$: depth(node i) (let root depth = 1)

$$\sum = 1$$

sorted keys:

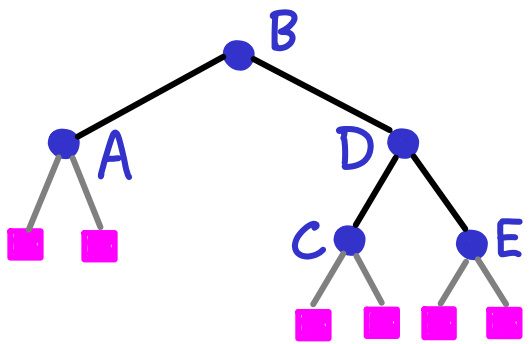
k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key:

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10



OPTIMAL BST : minimize expected search time

$E[\text{time}] =$

$\sum \text{prob}(i) \cdot \text{time}(i)$

$\text{prob}(i)$: p_i or q_i

$\text{time}(i)$: depth(node i) (let root depth = 1)

sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

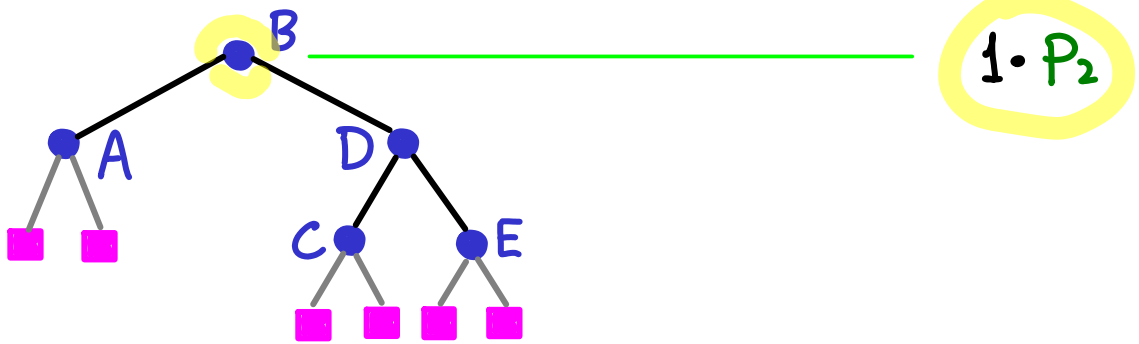
search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key:

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

$\sum = 1$



$1 \cdot p_2$

OPTIMAL BST : minimize expected search time

$E[\text{time}] = \sum \text{prob}(i) \cdot \text{time}(i)$

p_i or q_i

depth(node i)
(let root depth = 1)

$\sum = 1$ {

search probability:

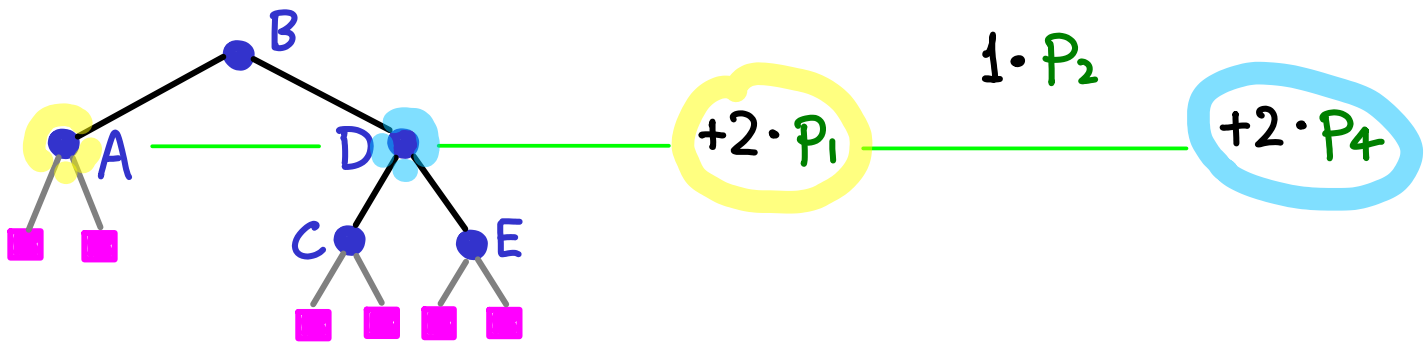
k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

sorted keys:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key :

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10



OPTIMAL BST : minimize expected search time

$E[\text{time}] =$

$\sum \text{prob}(i) \cdot \text{time}(i)$

p_i or q_i — depth(node i) (let root depth = 1)

sorted keys:

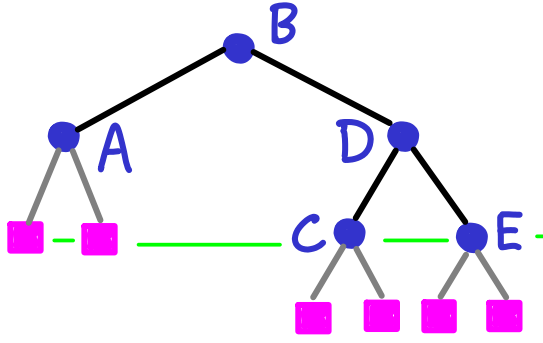
k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

$\sum = 1$ { search probability:

probability ≠ key :

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10



$1 \cdot p_2$

$+2 \cdot p_1$ $+2 \cdot p_4$

$+3 \cdot q_0$ $+3 \cdot q_1$ $+3 \cdot p_3$ $+3 \cdot p_5$

OPTIMAL BST : minimize expected search time

$E[\text{time}] =$

$\sum \text{prob}(i) \cdot \text{time}(i)$

p_i or q_i

depth(node i)
(let root depth = 1)

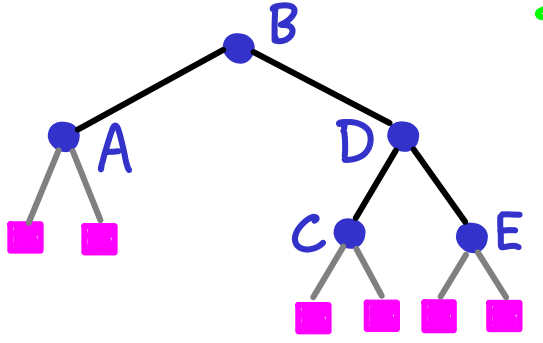
sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

$\sum = 1$ { search probability:
probability ≠ key :



$1 \cdot p_2$

$+2 \cdot p_1$ $+2 \cdot p_4$

$+3 \cdot q_0$ $+3 \cdot q_1$ $+3 \cdot p_3$ $+3 \cdot p_5$

$+4 \cdot q_2 + 4 \cdot q_3 + 4 \cdot q_4 + 4 \cdot q_5$

OPTIMAL BST : minimize expected search time

$E[\text{time}] = \sum \text{prob}(i) \cdot \text{time}(i)$

$\text{prob}(i)$: p_i or q_i

$\text{time}(i)$: depth(node i) (let root depth = 1)

sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

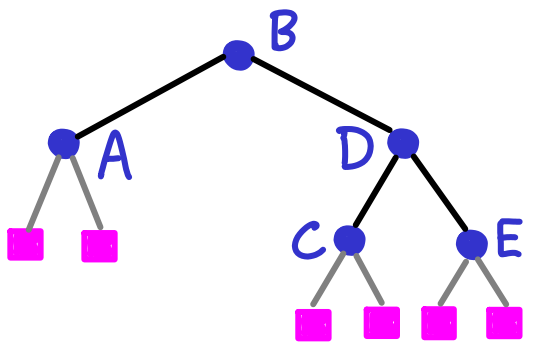
$\sum = 1$ {

search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key:

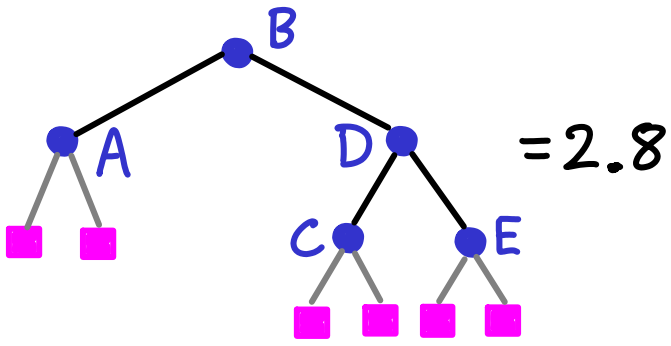
q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10



$1 \cdot p_2 + 2 \cdot p_1 + 2 \cdot p_4 + 3 \cdot q_0 + 3 \cdot p_3 + 3 \cdot q_1 + 3 \cdot p_5 + 4 \cdot q_2 + 4 \cdot q_3 + 4 \cdot q_4 + 4 \cdot q_5$

$.1 + .3 + .2 + .15 + .3 + .15 + .6 + .2 + .2 + .2 + .4 = 2.8$

OPTIMAL BST : minimize expected search time



= 2.8

sorted keys:

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

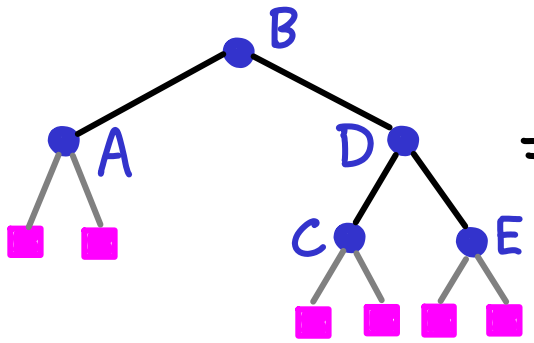
search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key :

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

OPTIMAL BST : minimize expected search time



$= 2.8$

sorted keys:

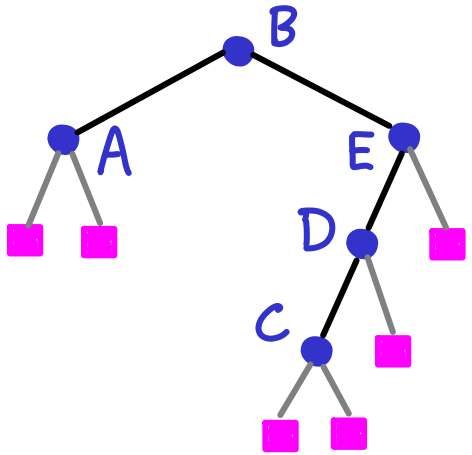
k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

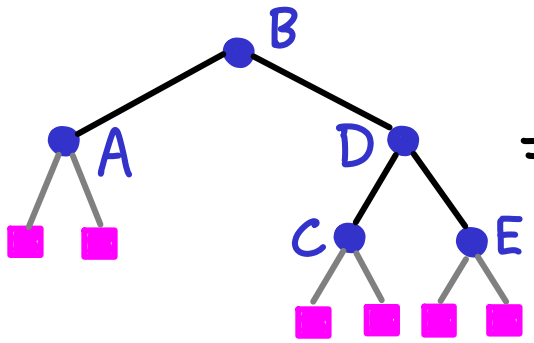
probability ≠ key :

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10



$$
\begin{aligned}
&1 \cdot .10 \\
&+ 2 \cdot (.15 + .20) \\
&+ 3 \cdot (.05 + .10 + .10 + .10) \\
&+ 4 \cdot (.05 + .05) \\
&+ 5 \cdot (.05 + .05)
\end{aligned}
\Bigg\} = 2.75
$$

OPTIMAL BST : minimize expected search time



$= 2.8$

sorted keys:

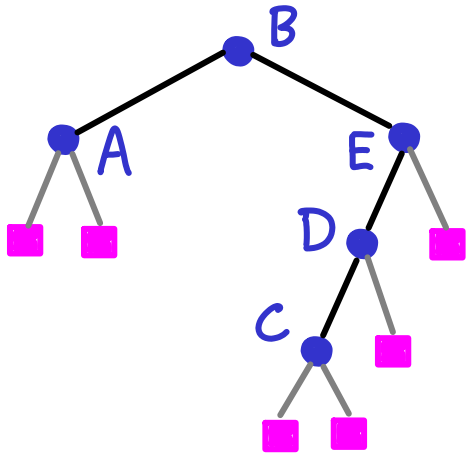
k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key :

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

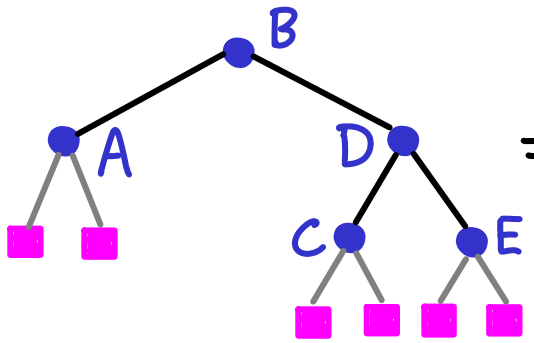


$$1 \cdot .10 + 2 \cdot (.15 + .20) + 3 \cdot (.05 + .10 + .10 + .10) + 4 \cdot (.05 + .05) + 5 \cdot (.05 + .05) = 2.75$$

We don't always get:

- balanced OPT BST

OPTIMAL BST : minimize expected search time



= 2.8

sorted keys:

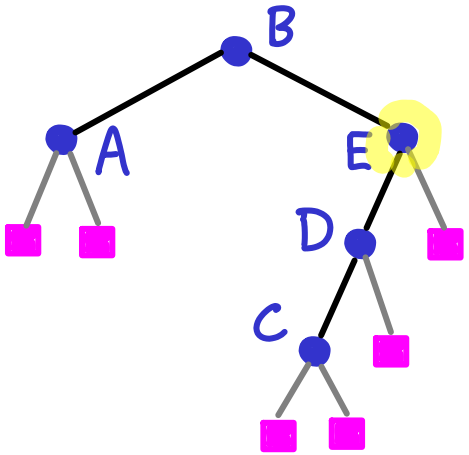
k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

search probability:

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

probability ≠ key :

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

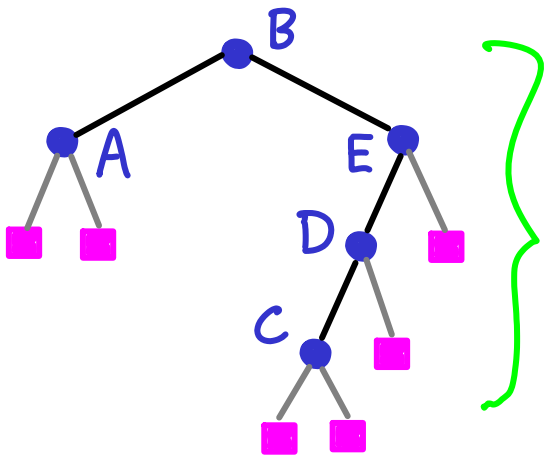


$$1 \cdot .10 + 2 \cdot (.15 + .20) + 3 \cdot (.05 + .10 + .10 + .10) + 4 \cdot (.05 + .05) + 5 \cdot (.05 + .05) = 2.75$$

We don't always get:

- balanced OPT BST
- max p_i at root

OPTIMAL BST : minimize expected search time



= 2.75

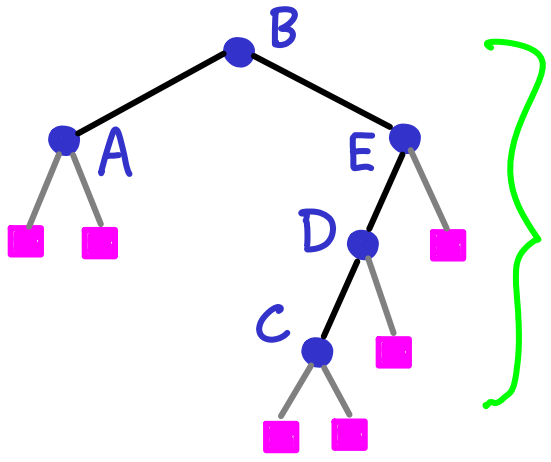
k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

We don't always get < balanced OPT BST / max p_i at root } not greedy

OPTIMAL BST : minimize expected search time



$= 2.75$

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

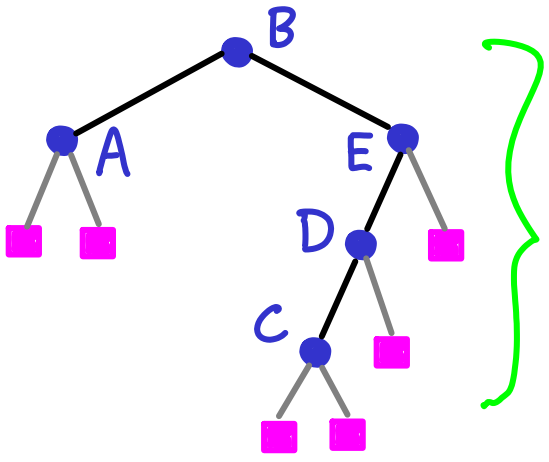
p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

We don't always get $<$ { balanced OPT BST / max p_i at root } not greedy

Try all trees? $\rightarrow \binom{2n}{n} \cdot \frac{1}{n+1} = \Omega\left(\frac{4^n}{n^{1.5}}\right)$ ☹

OPTIMAL BST : minimize expected search time



$= 2.75$

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

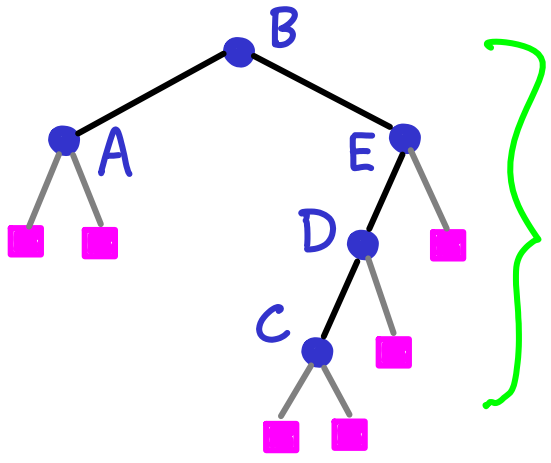
q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

We don't always get $<$ balanced OPT BST / max p_i at root } not greedy

Try all trees? $\rightarrow \binom{2n}{n} \cdot \frac{1}{n+1} = \Omega\left(\frac{4^n}{n^{1.5}}\right)$ ☹

...but an OPT-BST must contain OPT subtrees

OPTIMAL BST : minimize expected search time



$= 2.75$

k_1	k_2	k_3	k_4	k_5
A	B	C	D	E

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

We don't always get < balanced OPT BST / max p_i at root } not greedy

Try all trees? → $\binom{2n}{n} \cdot \frac{1}{n+1} = \Omega\left(\frac{4^n}{n^{1.5}}\right)$ ☹

...but an OPT-BST must contain OPT subtrees

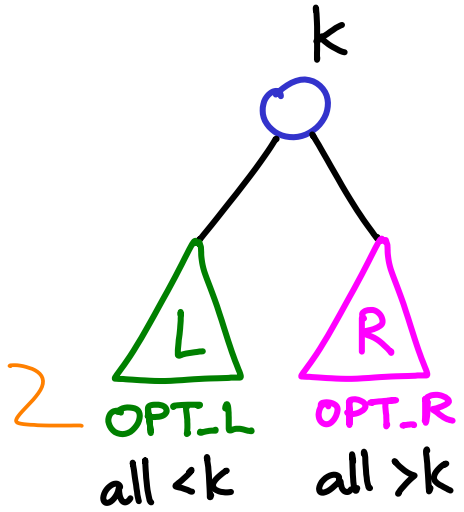
try DYNAMIC PROGRAMMING

↓

need to determine how to combine (few) subproblems

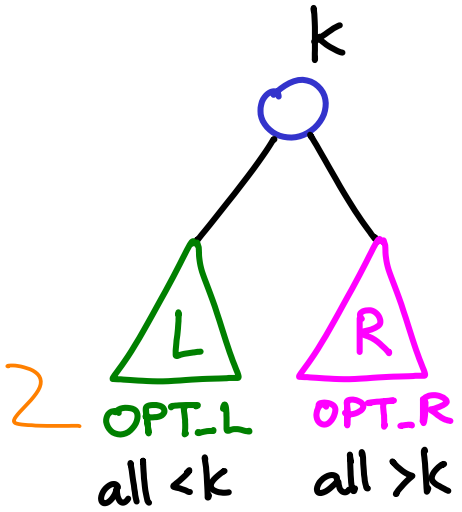
Suppose we fix a root, k

Then we know the contents of left & right subtrees, and we would like to be able to look up their score/structure → compute them first



Suppose we fix a root, k

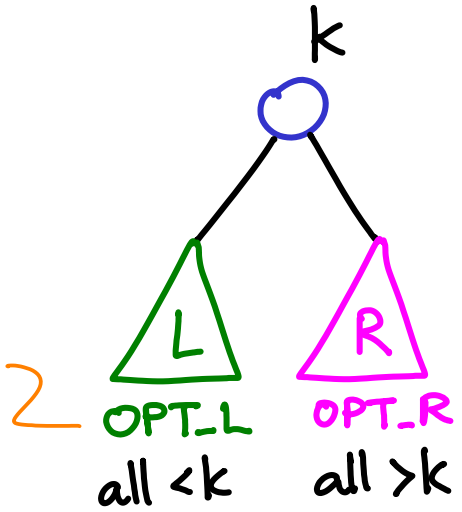
Then we know the contents of left & right subtrees, and we would like to be able to look up their score/structure → compute them first



How many subtrees / distinct subproblems overall?

Suppose we fix a root, k

Then we know the contents of left & right subtrees, and we would like to be able to look up their score/structure → compute them first

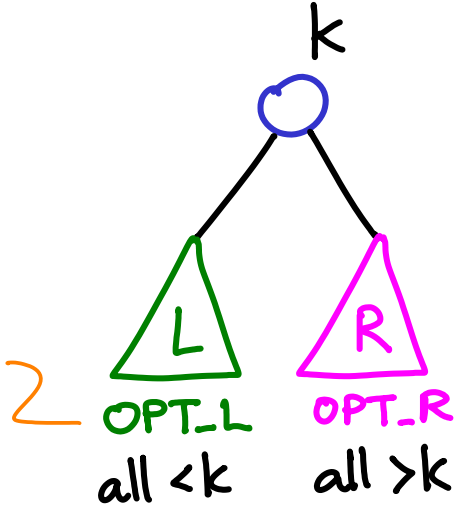


How many subtrees / distinct subproblems overall?

↳ base cases: empty trees → Q_i

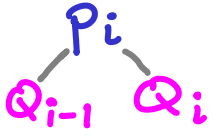
Suppose we fix a root, k

Then we know the contents of left & right subtrees, and we would like to be able to look up their score/structure → compute them first



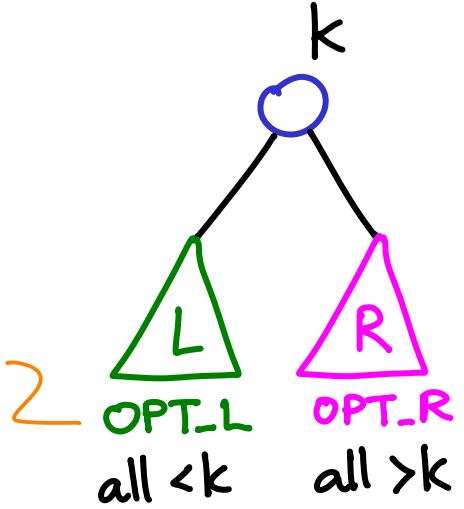
How many subtrees / distinct subproblems overall?

↳ base cases: empty trees → Q_i

↳ next level: singleton trees → 

Suppose we fix a root, k

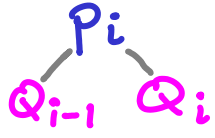
Then we know the contents of left & right subtrees, and we would like to be able to look up their score/structure → compute them first



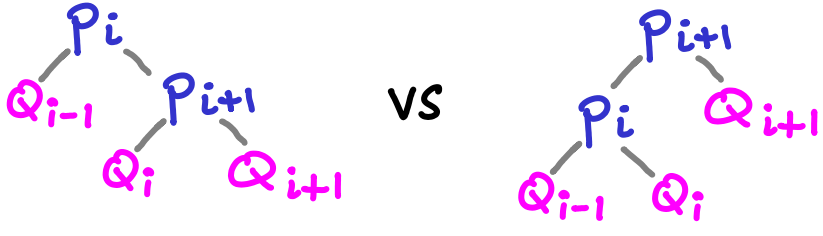
How many subtrees / distinct subproblems overall?

↳ base cases: empty trees → Q_i

↳ next level: singleton trees →

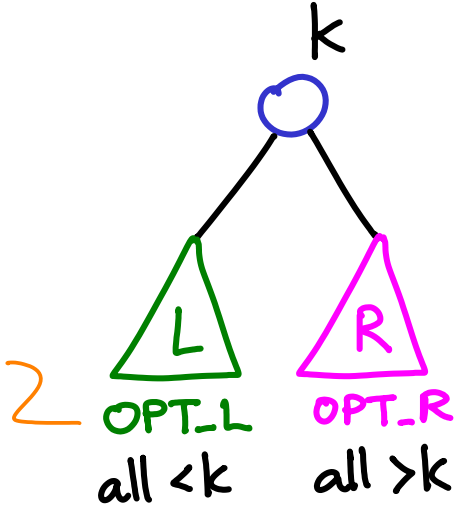


↳ next level: 2 consecutive keys →



Suppose we fix a root, k

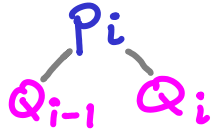
Then we know the contents of left & right subtrees, and we would like to be able to look up their score/structure → compute them first



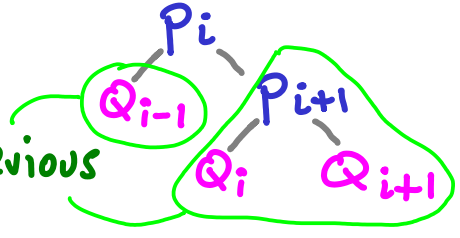
How many subtrees / distinct subproblems overall?

↳ base cases: empty trees → Q_i

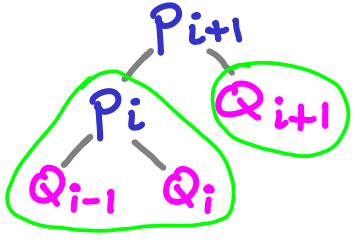
↳ next level: singleton trees →



↳ next level: 2 consecutive keys → relies on previous

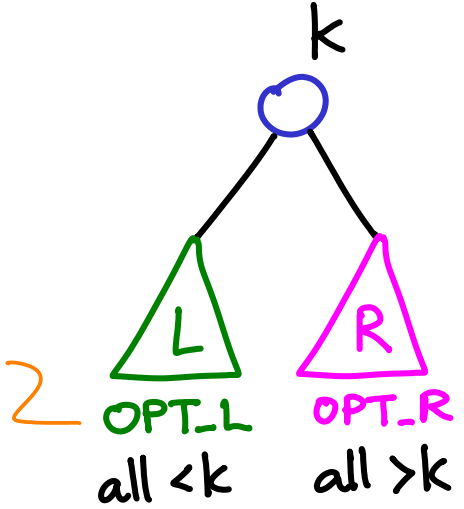


vs



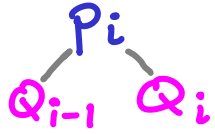
Suppose we fix a root, k

Then we know the contents of left & right subtrees, and we would like to be able to look up their score/structure → compute them first

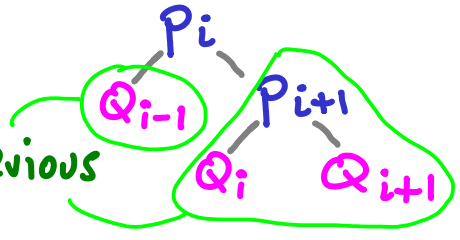


How many subtrees / distinct subproblems overall?

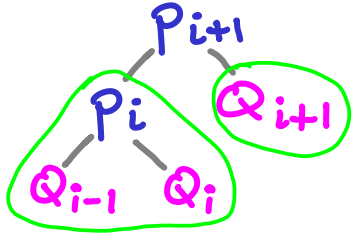
↳ base cases: empty trees → Q_i

↳ next level: singleton trees → 

↳ next level: 2 consecutive keys → relies on previous



vs

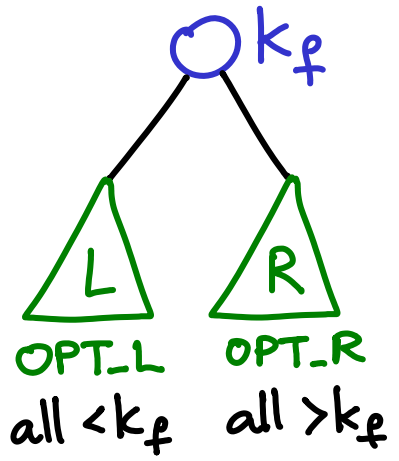


...

↳ all consecutive subsets $\sim 1 \le i \le j \le n$: $\Theta(n^2)$ ☺

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

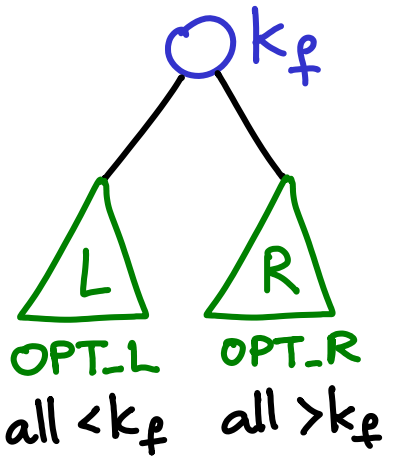
✓ & we have already computed OPT for all smaller subproblems



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

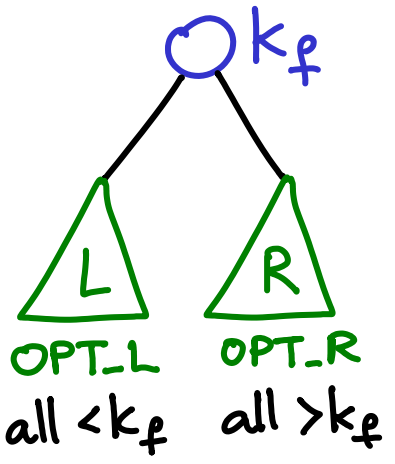
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



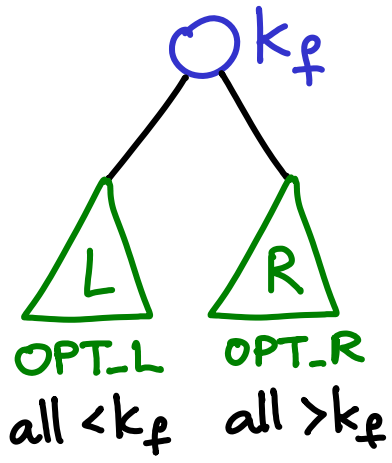
Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) =$?

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



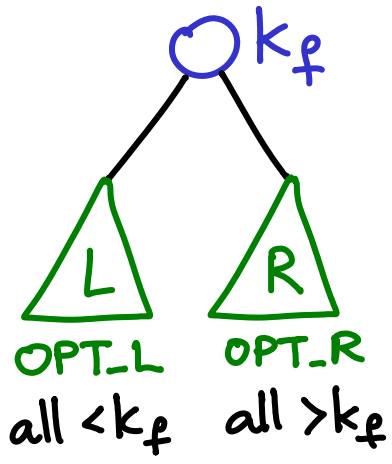
Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

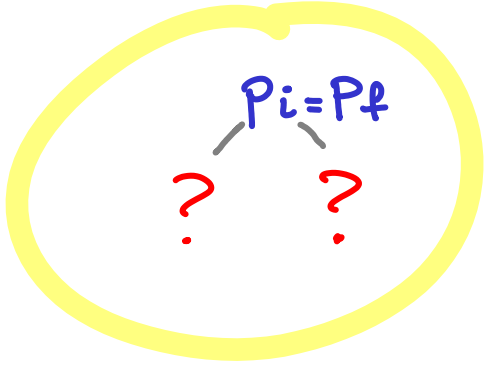
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

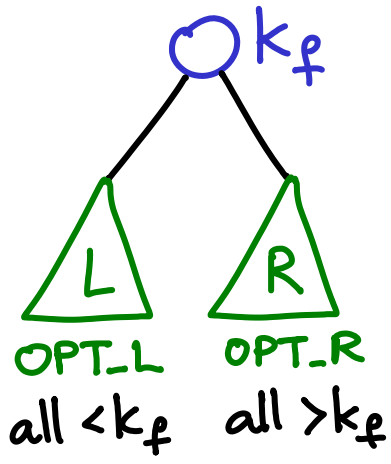
else if $f = i$, $e(f) =$?



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

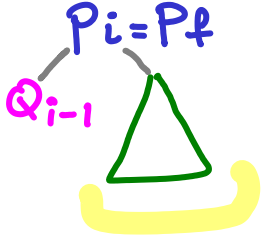
✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

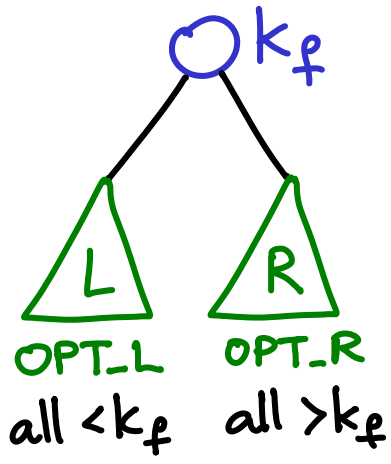


else if $f = i$, $e(f) = 2Q_{i-1} + p_f +$?

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

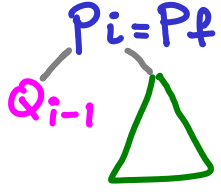
✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

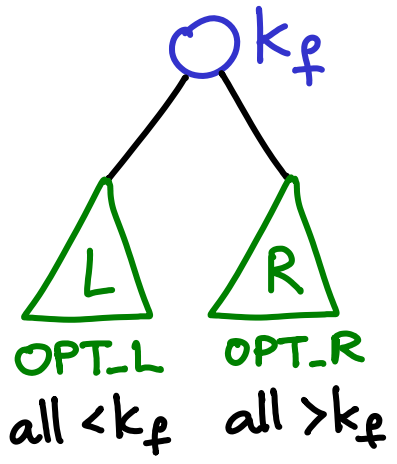


else if $f = i$, $e(f) = 2Q_{i-1} + p_f + OPT(f+1 \ldots j)$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

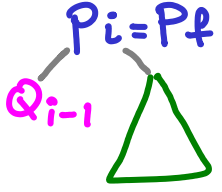


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

else if $f = i$, $e(f) = 2Q_{i-1} + p_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$

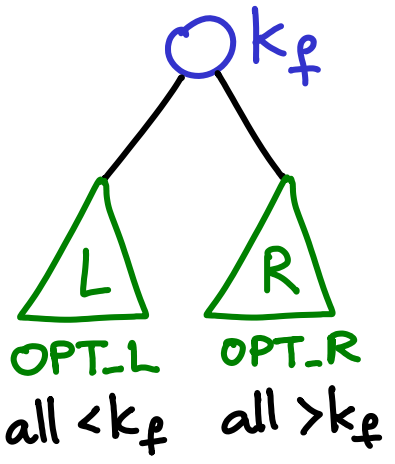
Why?



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

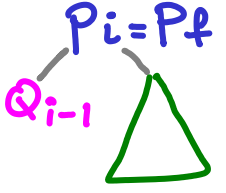


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

extra search time because right subtree got "lowered" by 1

if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$

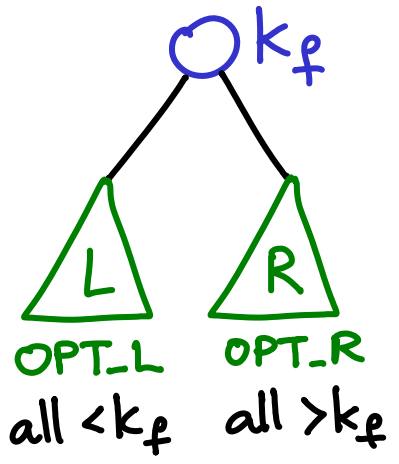
else if $f = i$, $e(f) = 2Q_{i-1} + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



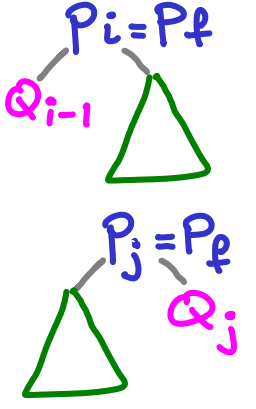
Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$

else if $f = i$, $e(f) = 2Q_{i-1} + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$

extra search time because right subtree got "lowered" by 1

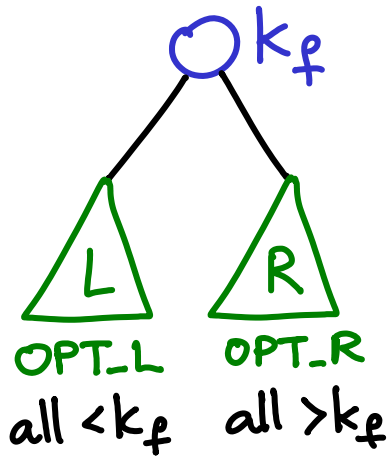
else if $f = j$,



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



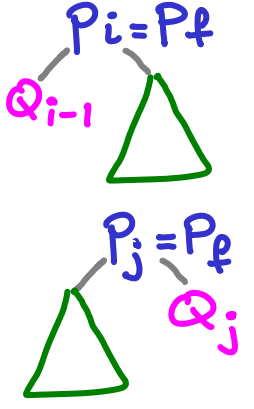
Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

extra search time because right subtree got "lowered" by 1

if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$

else if $f = i$, $e(f) = 2Q_{i-1} + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$

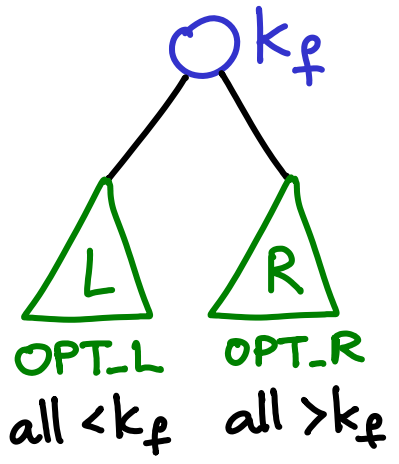
else if $f = j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + P_f + 2Q_j$



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

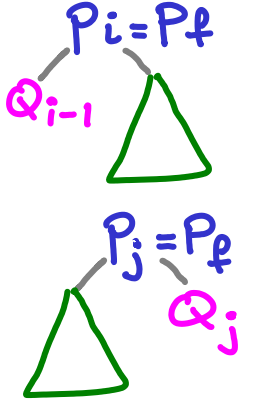
extra search time because right subtree got "lowered" by 1

if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$

else if $f = i$, $e(f) = 2Q_{i-1} + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$

else if $f = j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + P_f + 2Q_j$

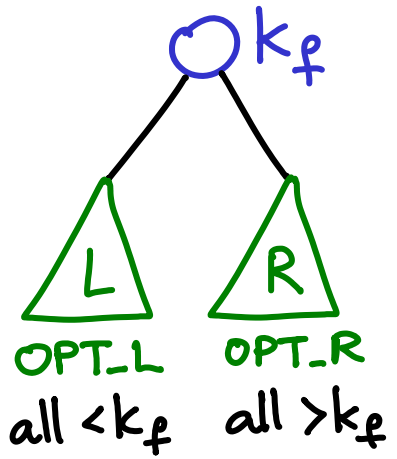
else $e(f) =$



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

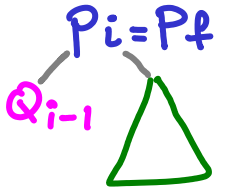


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

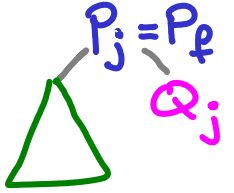
extra search time because right subtree got "lowered" by 1

if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$

else if $f = i$, $e(f) = 2Q_{i-1} + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$



else if $f = j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + P_f + 2Q_j$

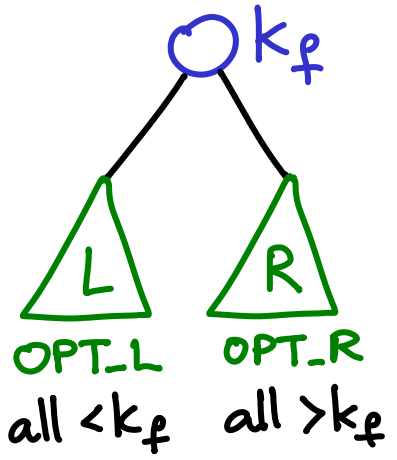


else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$



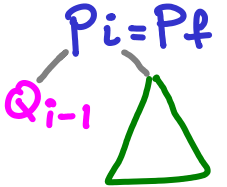
Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

W_L = weight of left child

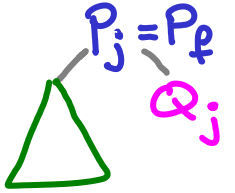
if $i=f=j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

↳ $Q_{i-1} + W_L + p_f + W_R + Q_j$

else if $f=i$, $e(f) = 2Q_{i-1} + p_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$



else if $f=j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + p_f + 2Q_j$

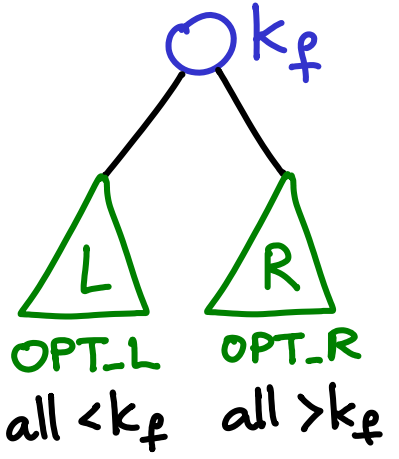


else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + p_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

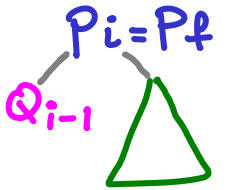


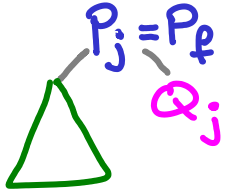
Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

W_f = weight of tree rooted at k_f
$= W_L + P_f + W_R$

if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$
↳ $Q_{i-1} + W_f + Q_j$

else if $f = i$, $e(f) = 2Q_{i-1} + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$





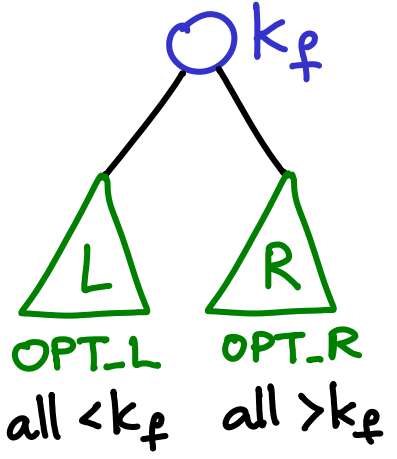
else if $f = j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + P_f + 2Q_j$

else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x) \right] + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x) \right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

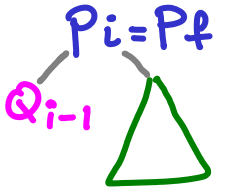


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

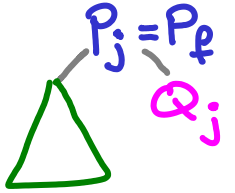
if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

↳ $Q_{i-1} + W_f + Q_j$

else if $f = i$, $e(f) = Q_{i-1} + W_L + p_f + [OPT(f+1 \ldots j) + W_R]$



else if $f = j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + p_f + 2Q_j$

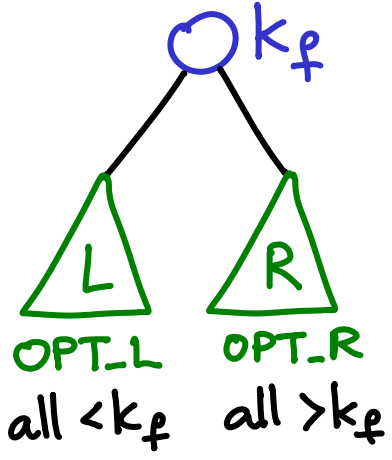


else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + p_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x)\right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

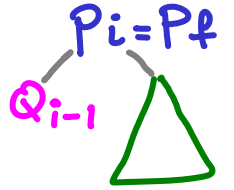


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

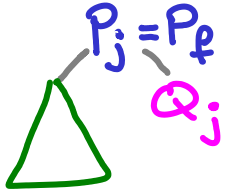
if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

↳ $Q_{i-1} + W_f + Q_j$

else if $f = i$, $e(f) = Q_{i-1} +$ W_f $+ OPT(f+1 \ldots j)$



else if $f = j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + p_f + 2Q_j$

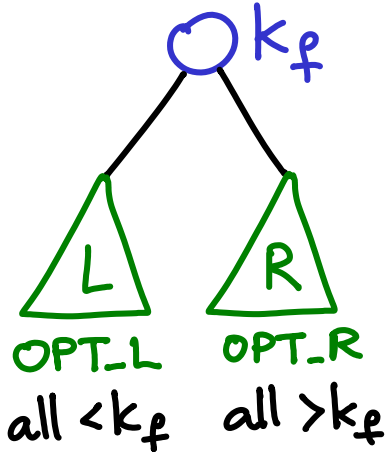


else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + p_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x)\right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

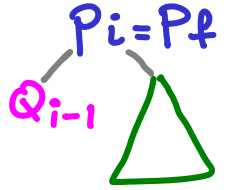


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

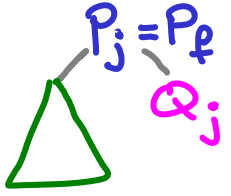
if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

↳ $Q_{i-1} + W_f + Q_j$

else if $f = i$, $e(f) = Q_{i-1} + W_f + OPT(f+1 \ldots j)$



else if $f = j$, $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + p_f + 2Q_j$

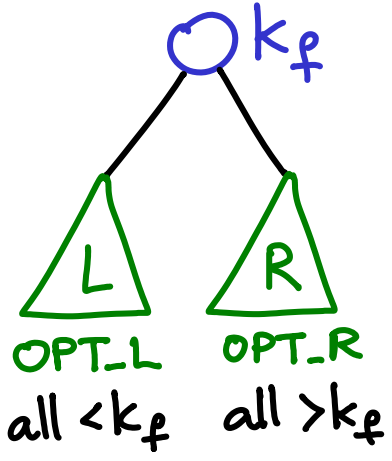


else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + p_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x)\right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

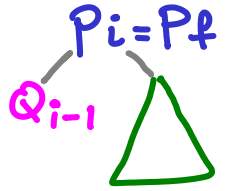


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

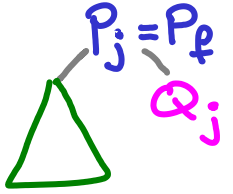
if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$

↳ $Q_{i-1} + W_f + Q_j$

else if $f = i$, $e(f) = Q_{i-1} + W_f + OPT(f+1 \ldots j)$



else if $f = j$, $e(f) = OPT(i \ldots f-1) + W_f + Q_j$

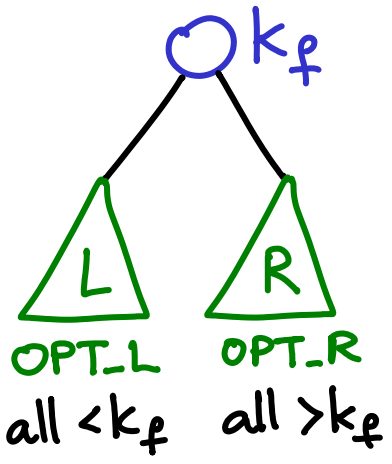


else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + P_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x)\right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

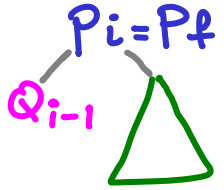


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

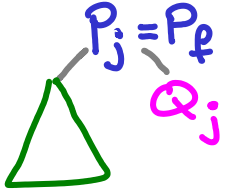
if $i = f = j$, $e(f) = 2Q_{i-1} + p_f + 2Q_j$

↳ $Q_{i-1} + W_f + Q_j$

else if $f = i$, $e(f) = Q_{i-1} + W_f + OPT(f+1 \ldots j)$



else if $f = j$, $e(f) = OPT(i \ldots f-1) + W_f + Q_j$

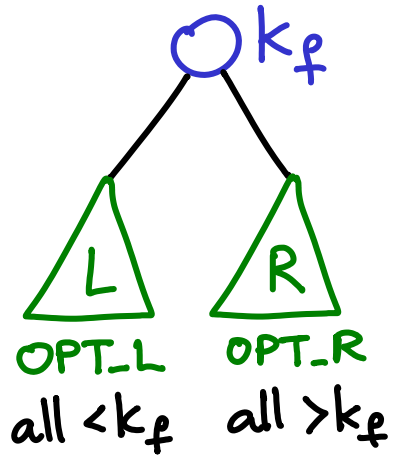


else $e(f) = \left[OPT(i \ldots f-1) + \sum_{x=i}^{f-1} prob(x)\right] + p_f + \left[OPT(f+1 \ldots j) + \sum_{x=f+1}^{j} prob(x)\right]$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

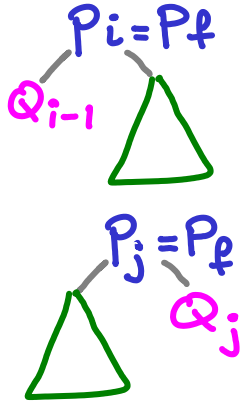


Recall: $E[\text{time}] = \sum \text{depth} \cdot \text{prob}$

if $i = f = j$, $e(f) = 2Q_{i-1} + P_f + 2Q_j$

↳ $Q_{i-1} + W_f + Q_j$

else if $f = i$, $e(f) = Q_{i-1} + W_f + OPT(f+1 \ldots j)$



else if $f = j$, $e(f) = OPT(i \ldots f-1) + W_f + Q_j$

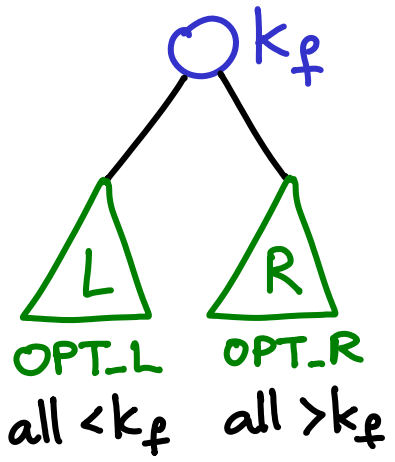
else $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$



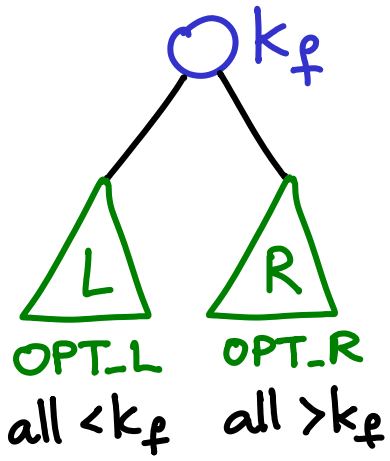
Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

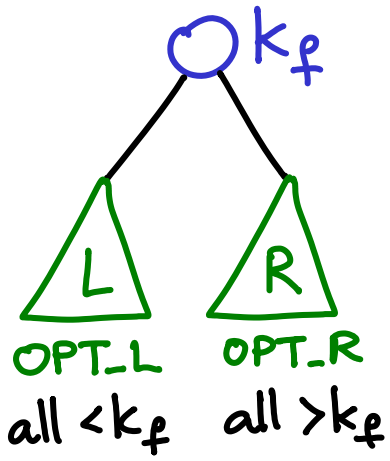
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

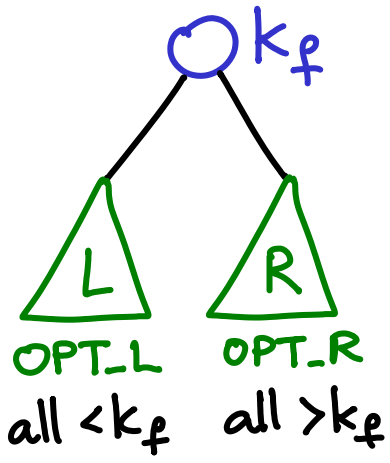
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Finally, we will try out all possible roots: (for $f = i$ to j)

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

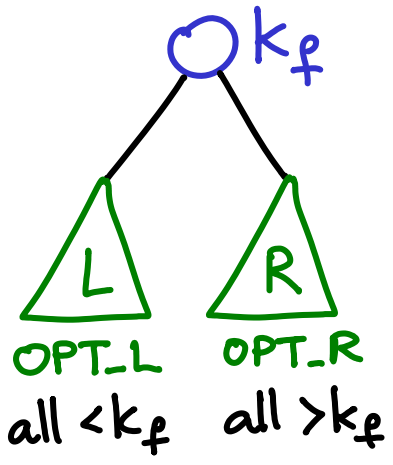
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Finally, we will try out all possible roots: (for $f = i$ to j)

Time complexity: ?

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

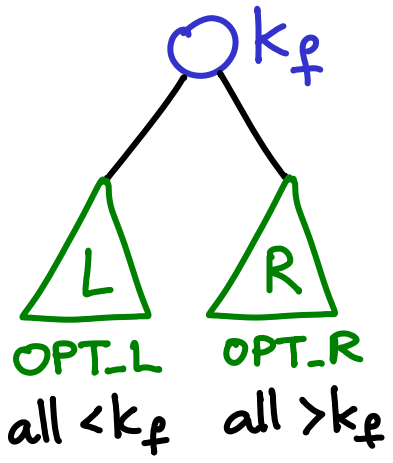
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Finally, we will try out all possible roots: (for f = i to j)

Time complexity: - e(f) costs ?

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

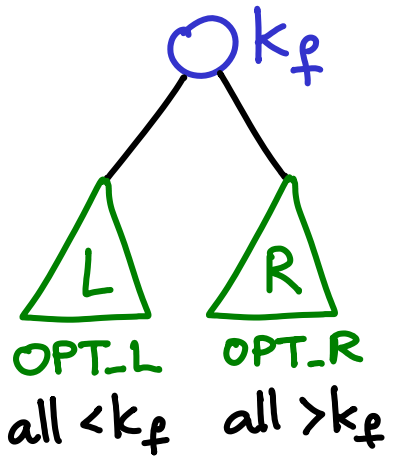
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Finally, we will try out all possible roots: (for f = i to j)

Time complexity: - e(f) costs O(1) - all terms are already computed

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

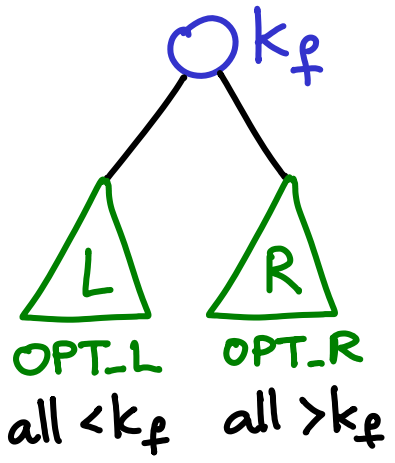
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Finally, we will try out all possible roots: (for f = i to j)

Time complexity:

- $e(f)$ costs $O(1)$ - all terms are already computed
- $OPT(i \ldots j)$ costs $O(j-i) = O(n)$ - try all roots

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

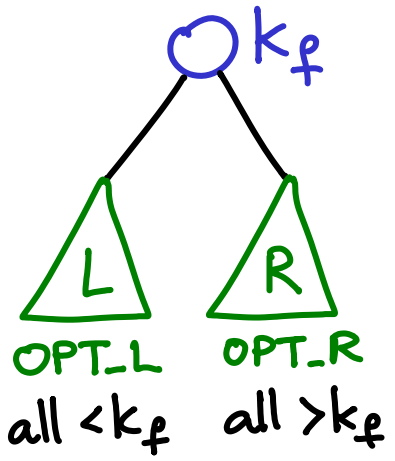
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Finally, we will try out all possible roots: (for $f = i$ to j)

Time complexity:

- $e(f)$ costs $O(1)$ - all terms are already computed
- $OPT(i \ldots j)$ costs $O(j-i) = O(n)$ - try all roots
- $\Theta(n^2)$ distinct OPT values

Suppose we fix a root, k_f for subproblem $(i \ldots j)$ // $i \le f \le j$

✓ & we have already computed OPT for all smaller subproblems

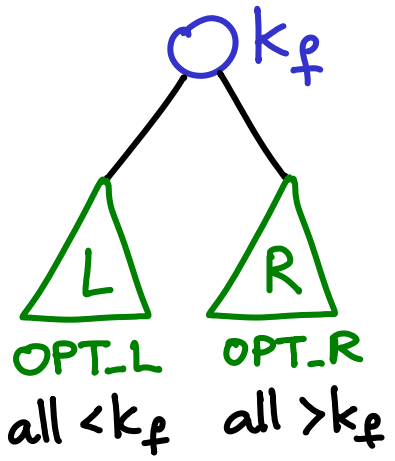
How do we get $E[\text{for fixed } k_f]$? // $= e(f)$

↳ In general $e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

✓ ✓

need all these

↳ will be easy



Finally, we will try out all possible roots: (for $f = i$ to j)

Time complexity:

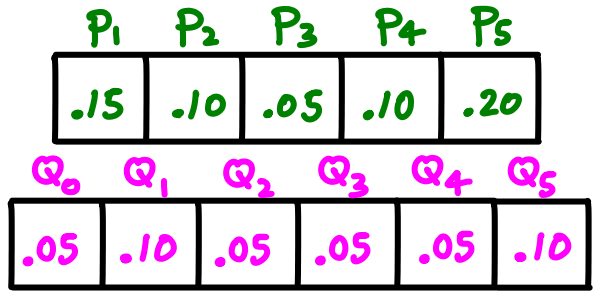
$\Theta(n^3)$

- $e(f)$ costs $O(1)$ - all terms are already computed
- $OPT(i \ldots j)$ costs $O(j-i) = O(n)$ - try all roots
- $\Theta(n^2)$ distinct OPT values

$$e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$$

need all these

(total probability in subtree containing $k_i \ldots k_j$, with root k_f)



$e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

need all these

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include q_{i-1} & q_j

$e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

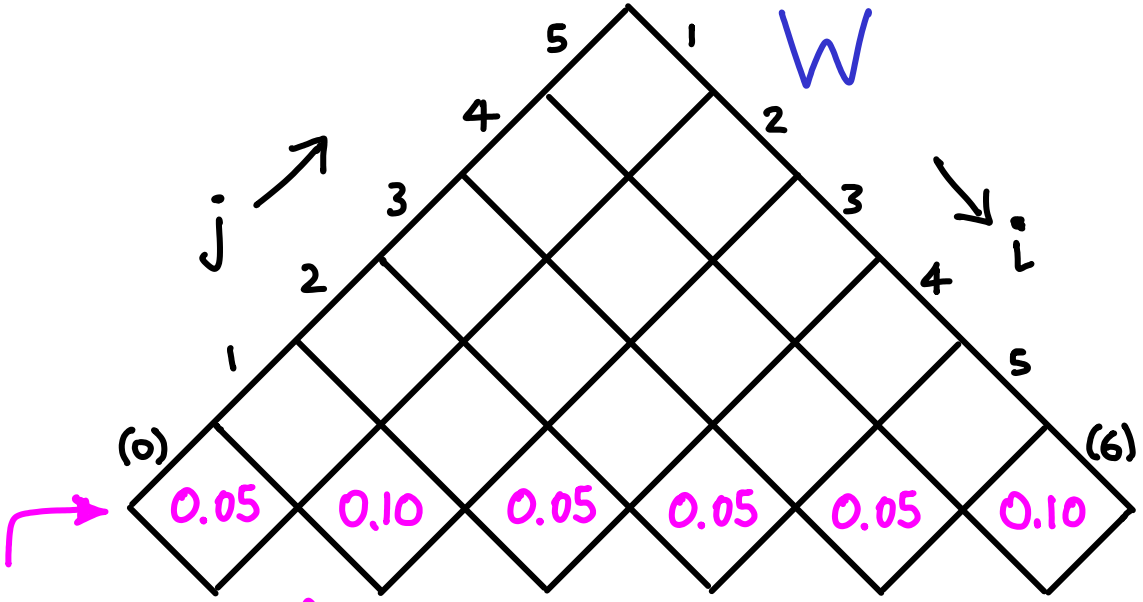
need all these

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j



base cases for empty trees

$e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

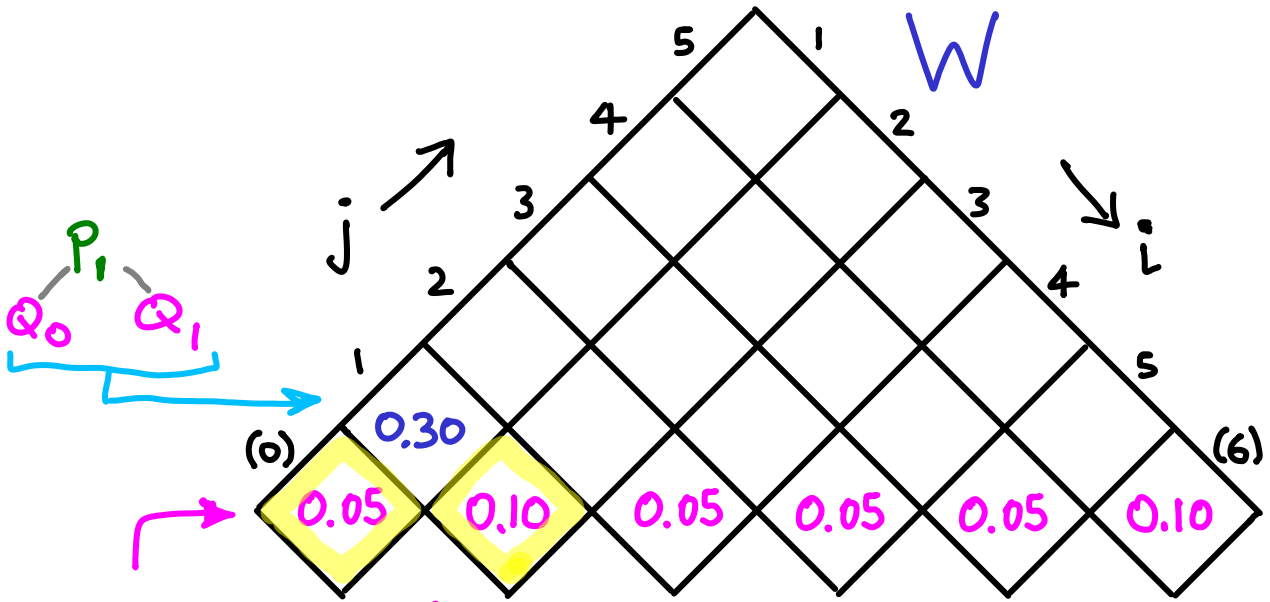
need all these

P_1	P_2	P_3	P_4	P_5
.15	.10	.05	.10	.20

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
.05	.10	.05	.05	.05	.10

(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j



base cases for empty trees

$$e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$$

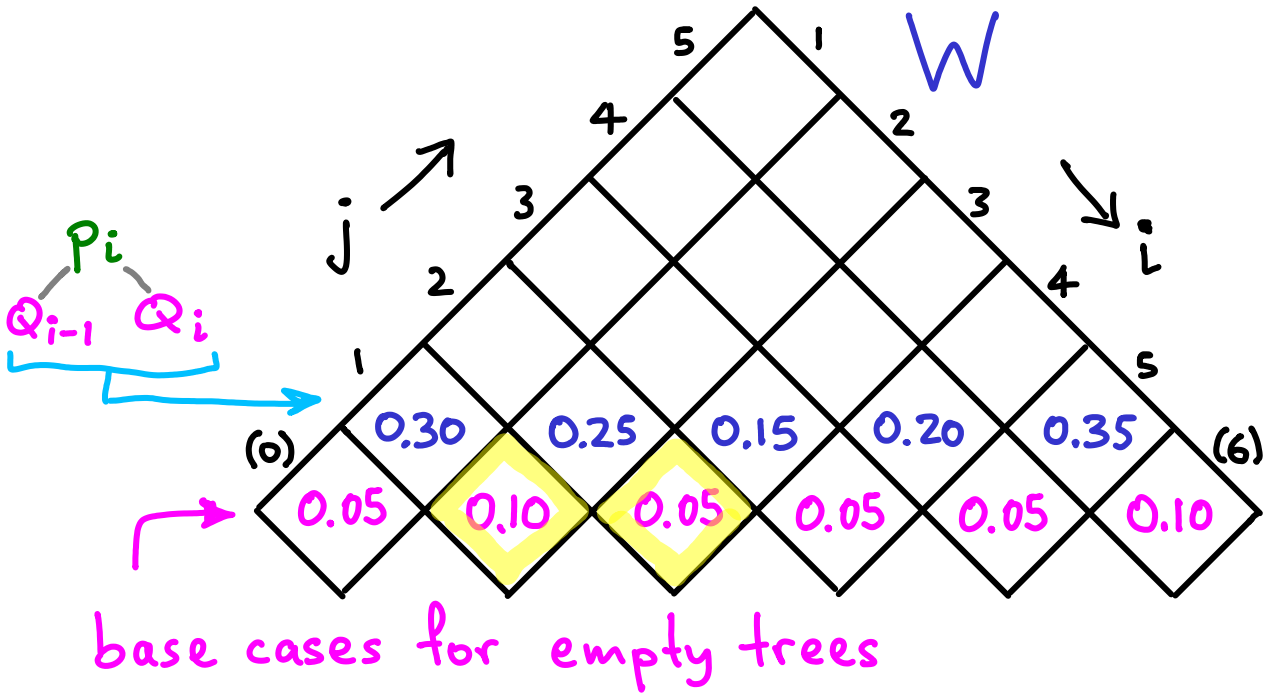
need all these

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j



base cases for empty trees

$e(f) = \text{OPT}(i \ldots f-1) + W_f + \text{OPT}(f+1 \ldots j)$

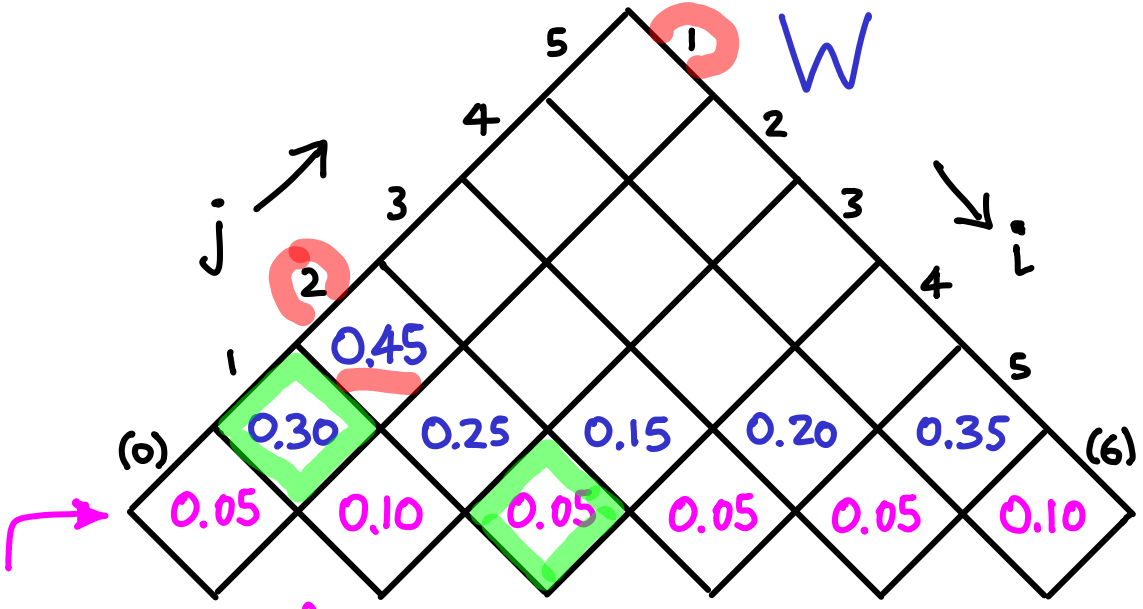
need all these

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

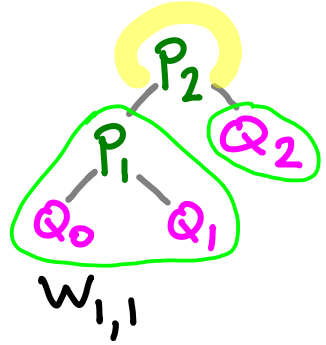
(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j

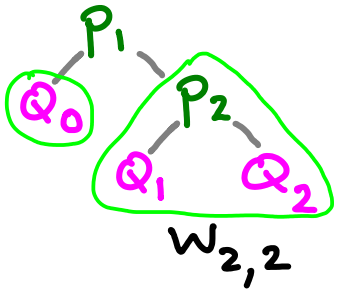


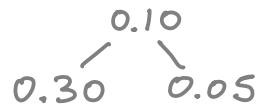
base cases for empty trees

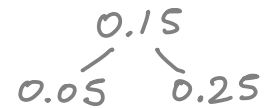
$W_{1,2} =$



$=$







$e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

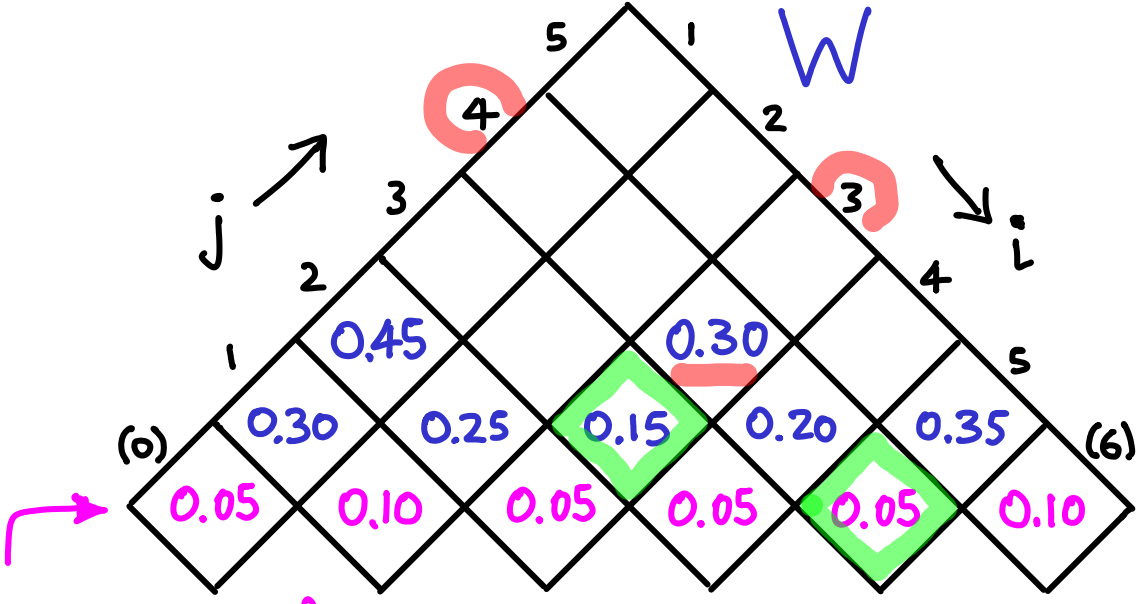
need all these

P_1	P_2	P_3	P_4	P_5
.15	.10	.05	.10	.20

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
.05	.10	.05	.05	.05	.10

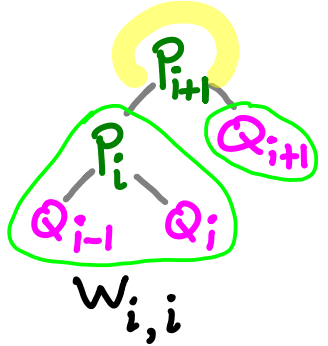
(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j



base cases for empty trees

$W_{i,i+1} =$



$e(f) = \text{OPT}(i \ldots f-1) + W_f + \text{OPT}(f+1 \ldots j)$

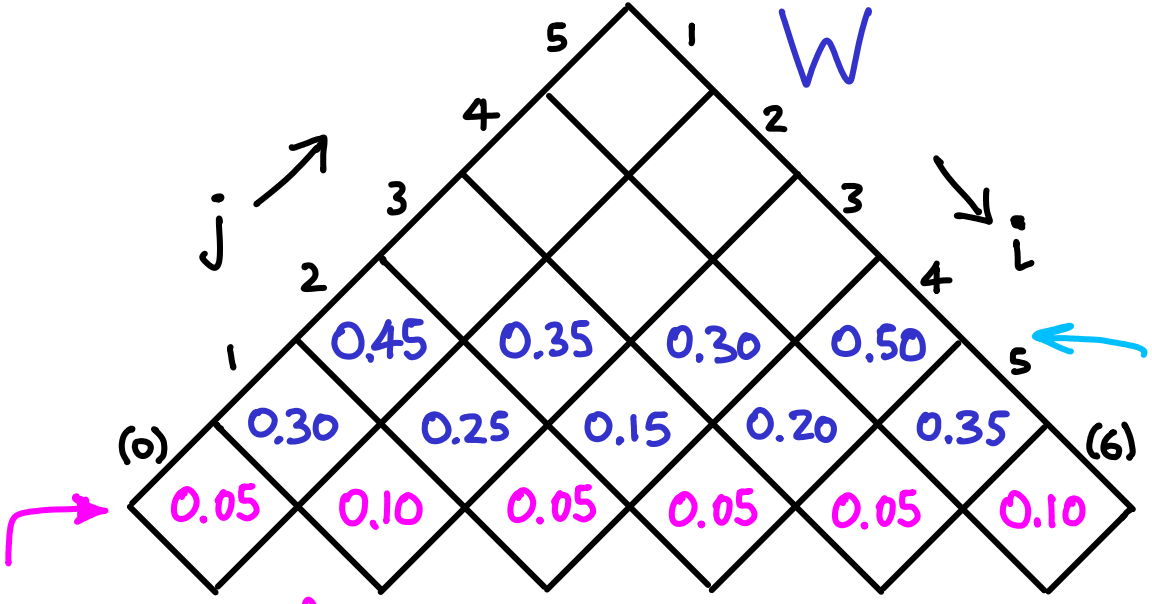
need all these

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

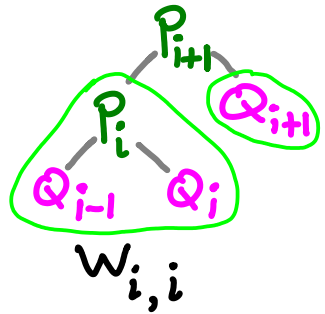
(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j



base cases for empty trees

$W_{i,i+1} =$



$e(f) = \text{OPT}(i \ldots f-1) + W_f + \text{OPT}(f+1 \ldots j)$

↓

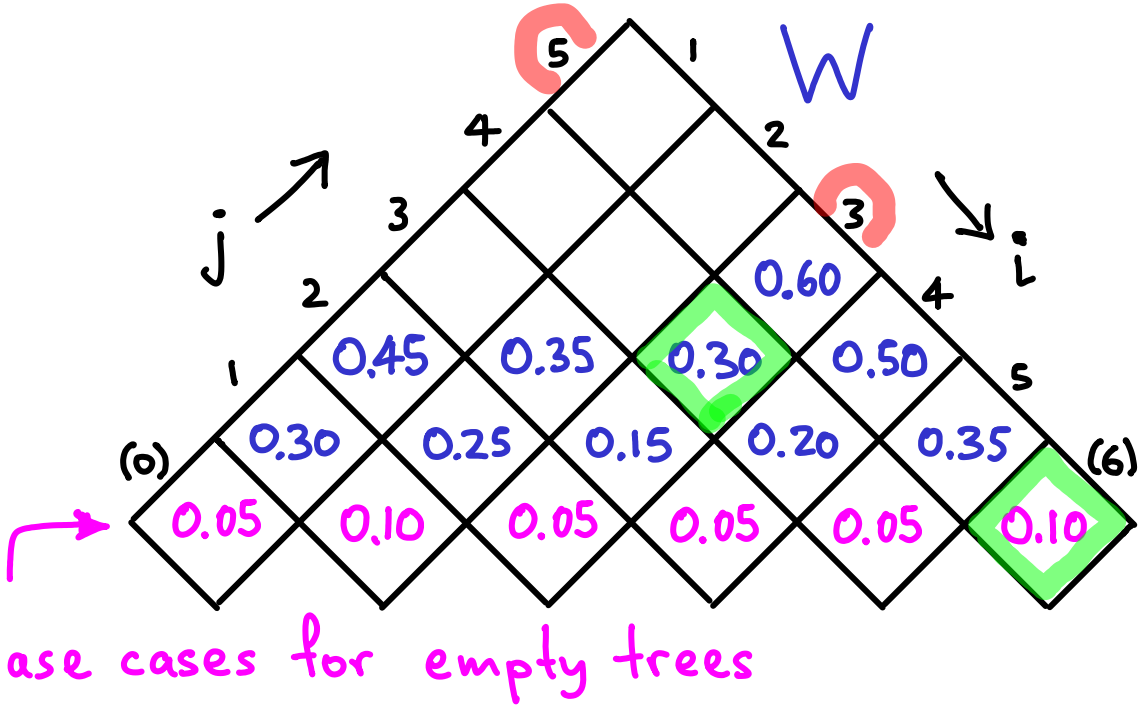
need all these

P_1	P_2	P_3	P_4	P_5
.15	.10	.05	.10	.20

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
.05	.10	.05	.05	.05	.10

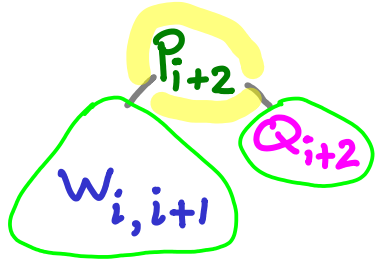
(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j



base cases for empty trees

$W_{i,i+2} =$



$e(f) = OPT(i \ldots f-1) + W_f + OPT(f+1 \ldots j)$

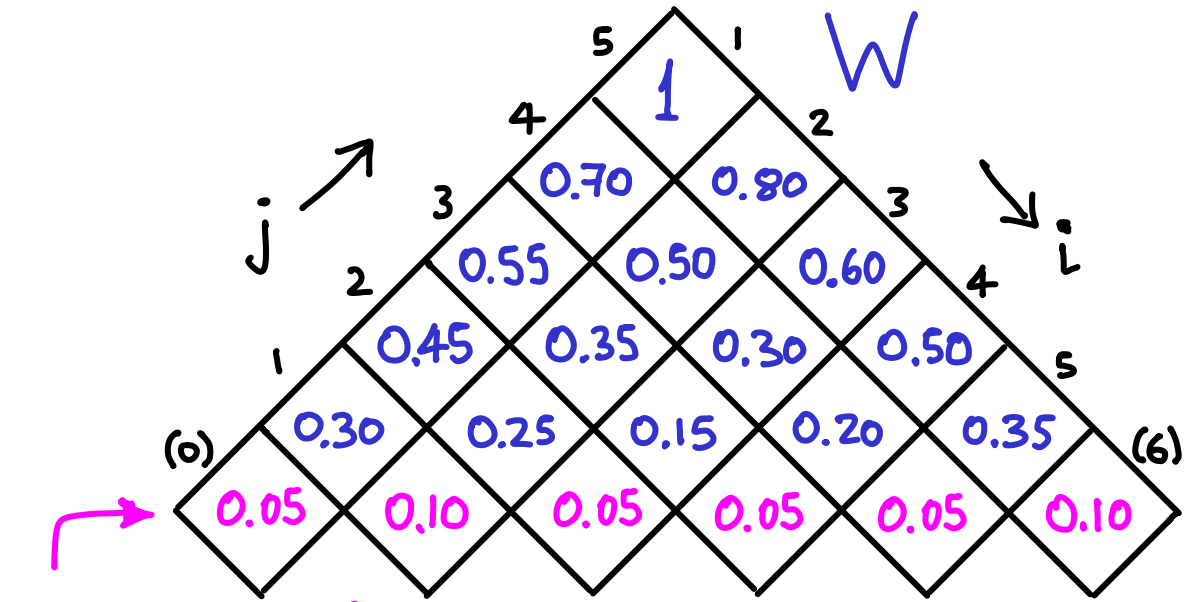
need all these

p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

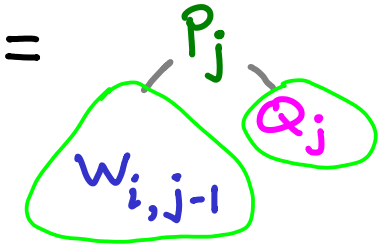
q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

(total probability in subtree containing $k_i \ldots k_j$, with root k_f)

↳ but it's independent of root: only i & j matter // must include Q_{i-1} & Q_j



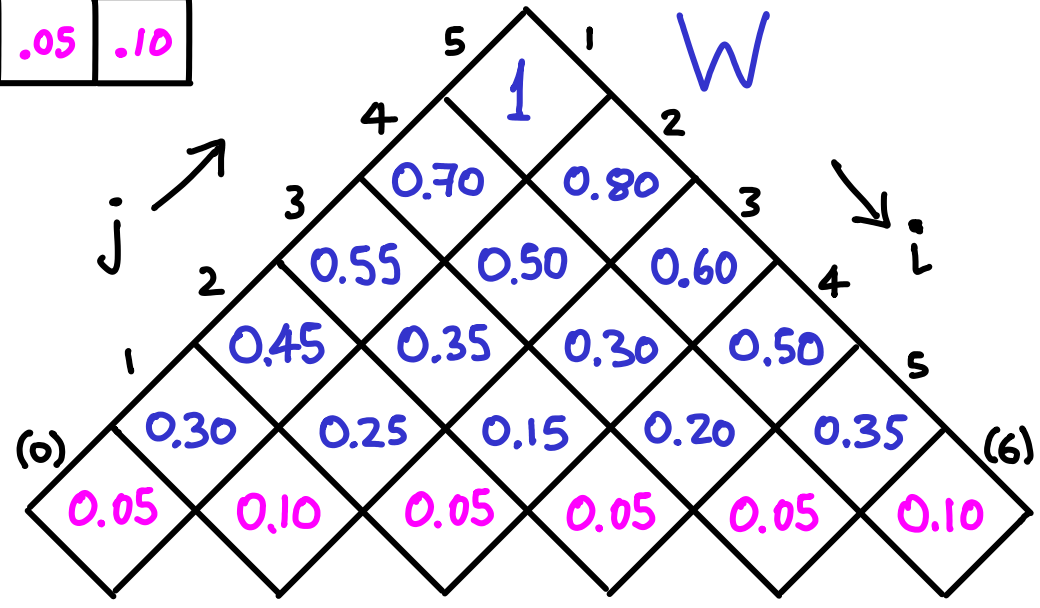
base cases for empty trees

$W_{i,j} =$ 

Build W table: $\Theta(n^2)$

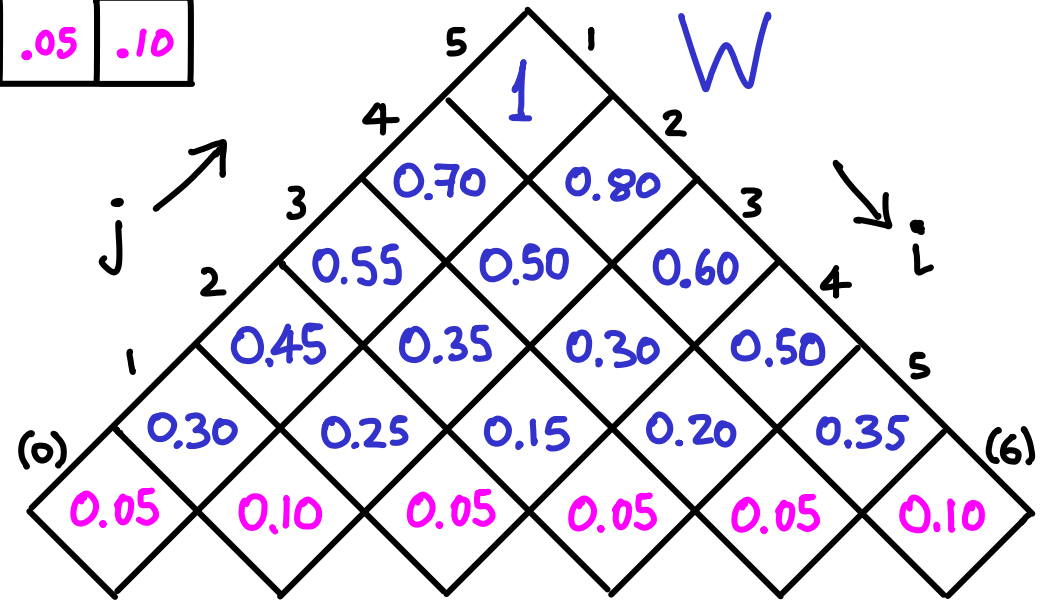
P_1	P_2	P_3	P_4	P_5
.15	.10	.05	.10	.20

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
.05	.10	.05	.05	.05	.10



p_1	p_2	p_3	p_4	p_5
.15	.10	.05	.10	.20

q_0	q_1	q_2	q_3	q_4	q_5
.05	.10	.05	.05	.05	.10

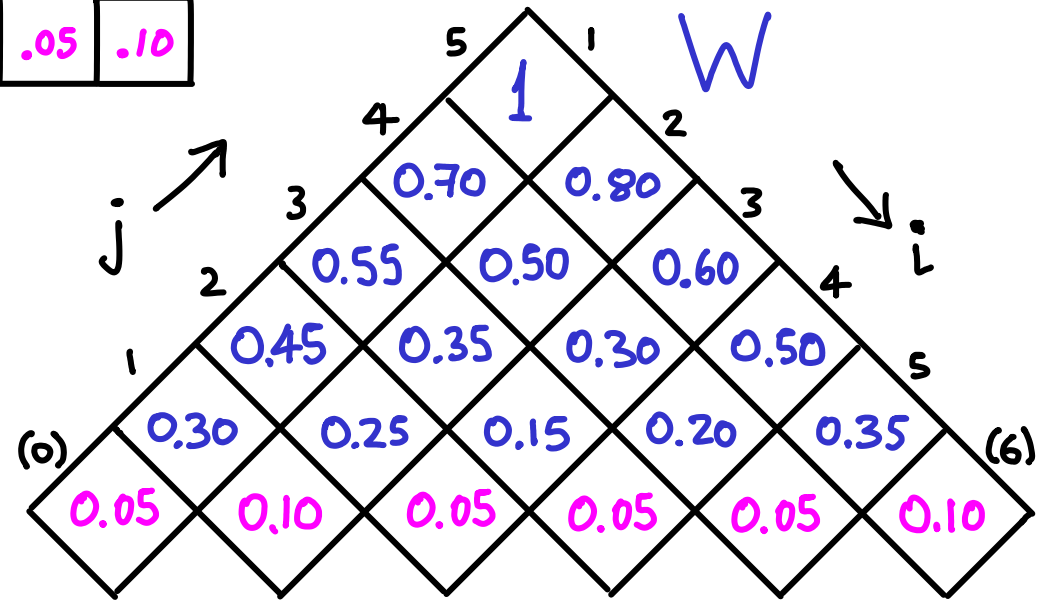


$$OPT(i,j) = \min_{\substack{\text{all roots} \\ f \text{ in } i \ldots j}} \{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \}$$

↳ was W_f but only i,j matter

P_1	P_2	P_3	P_4	P_5
.15	.10	.05	.10	.20

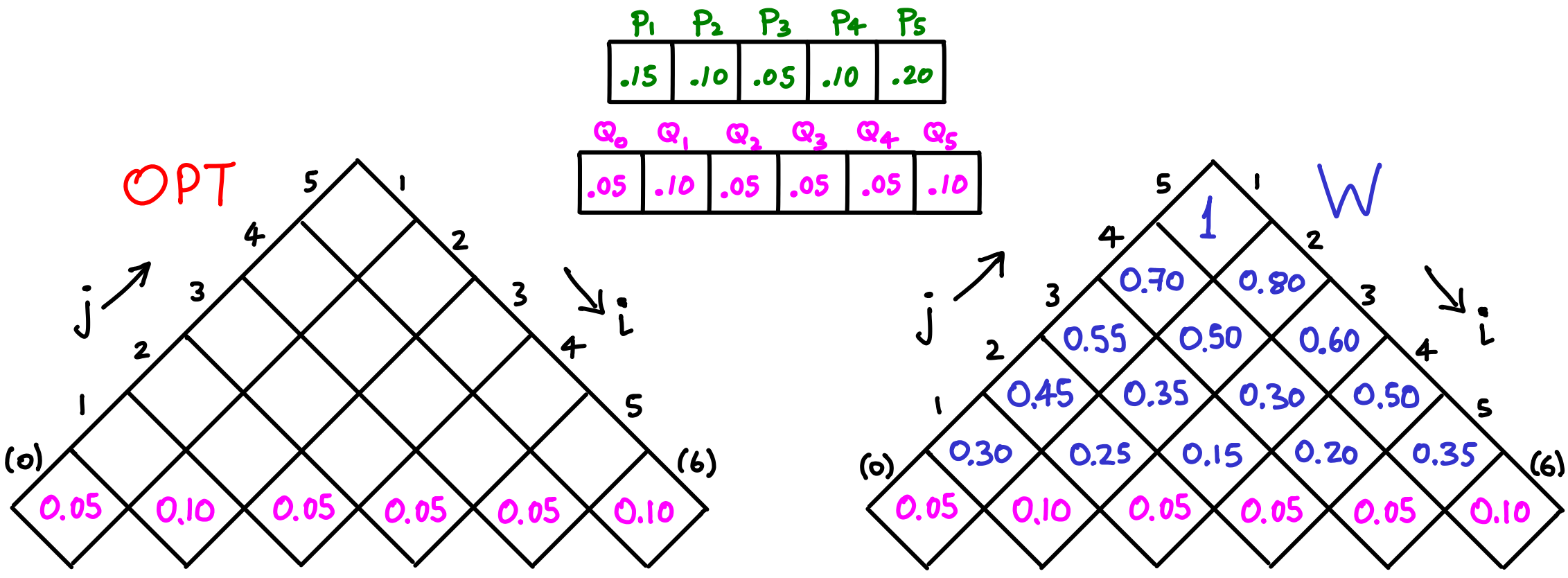
Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
.05	.10	.05	.05	.05	.10



$$OPT(i,j) = \min_{\text{all roots } f \text{ in } i\ldots j} \{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \}$$

Recall: if $i=f$ this is Q_i

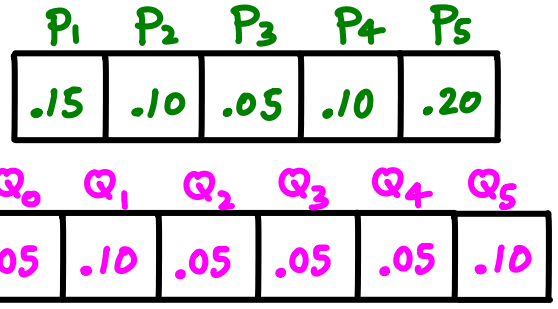
if $f=j$ this is Q_j

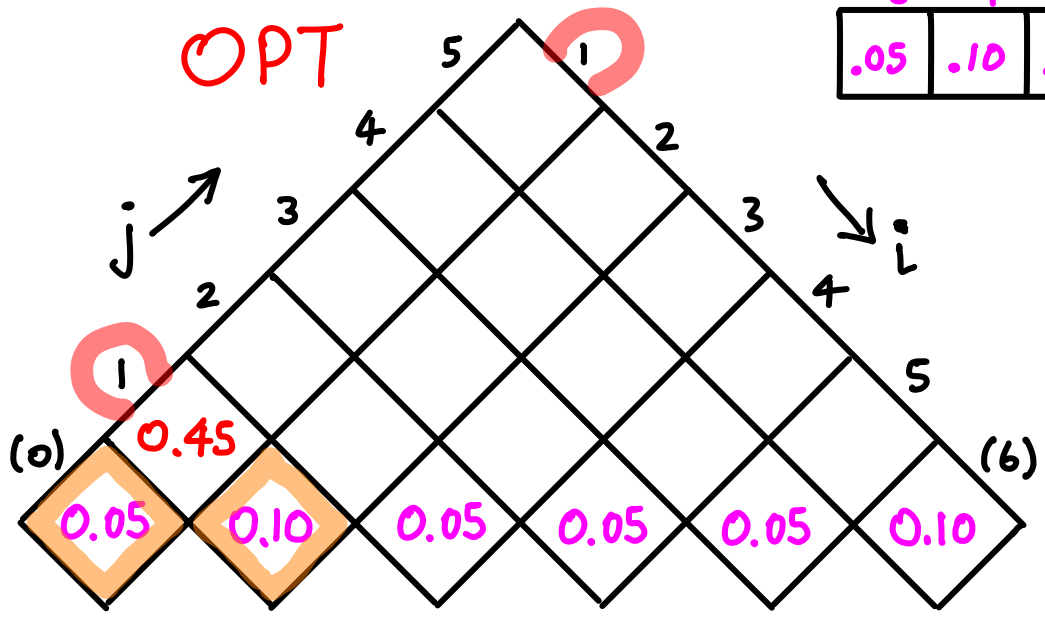


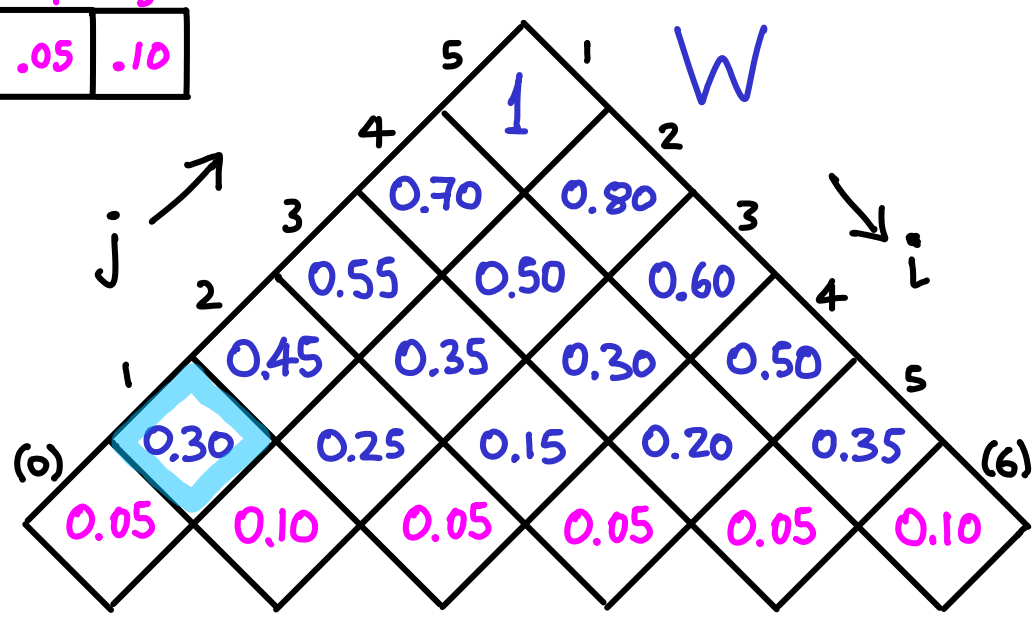
$$OPT(i,j) = \underset{\text{all roots } f \text{ in } i\ldots j}{\text{MIN}} \left\{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \right\}$$

Recall: if $i=f$ this is Q_i

if $f=j$ this is Q_j



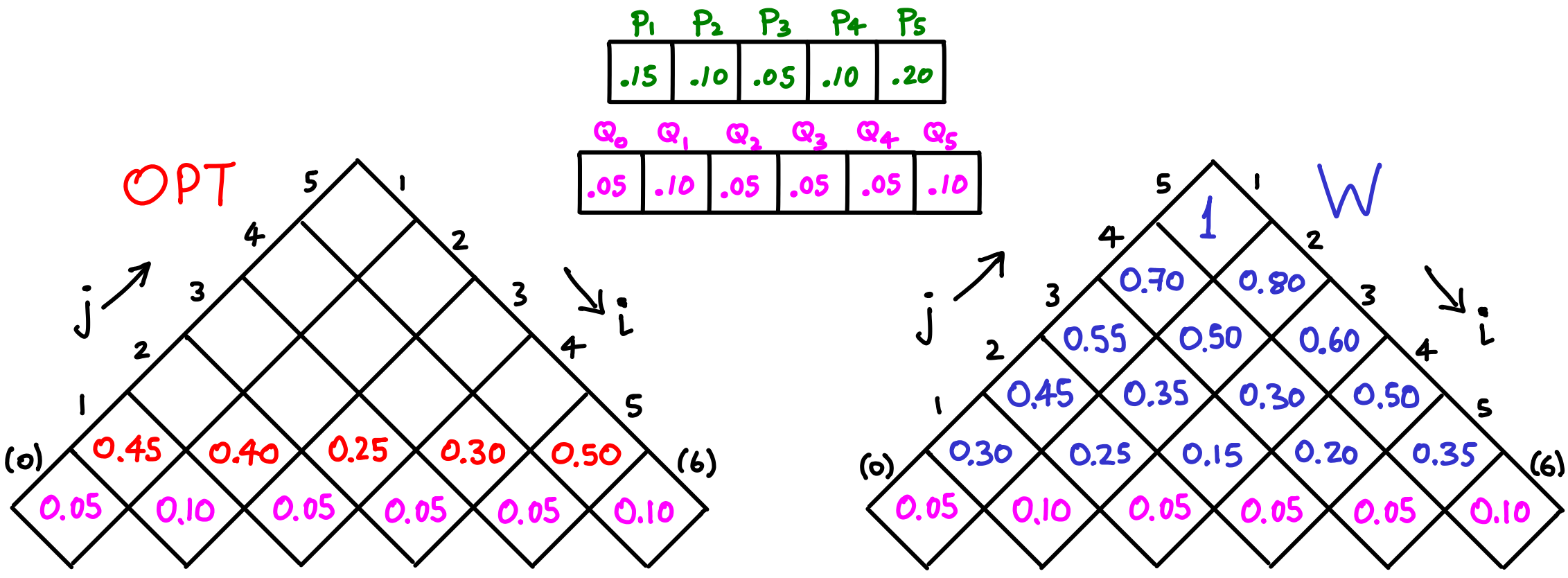




$$OPT(i,j) = \min_{\text{all roots } f \text{ in } i\ldots j} \{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \}$$

Recall: if $i=f$ this is Q_i

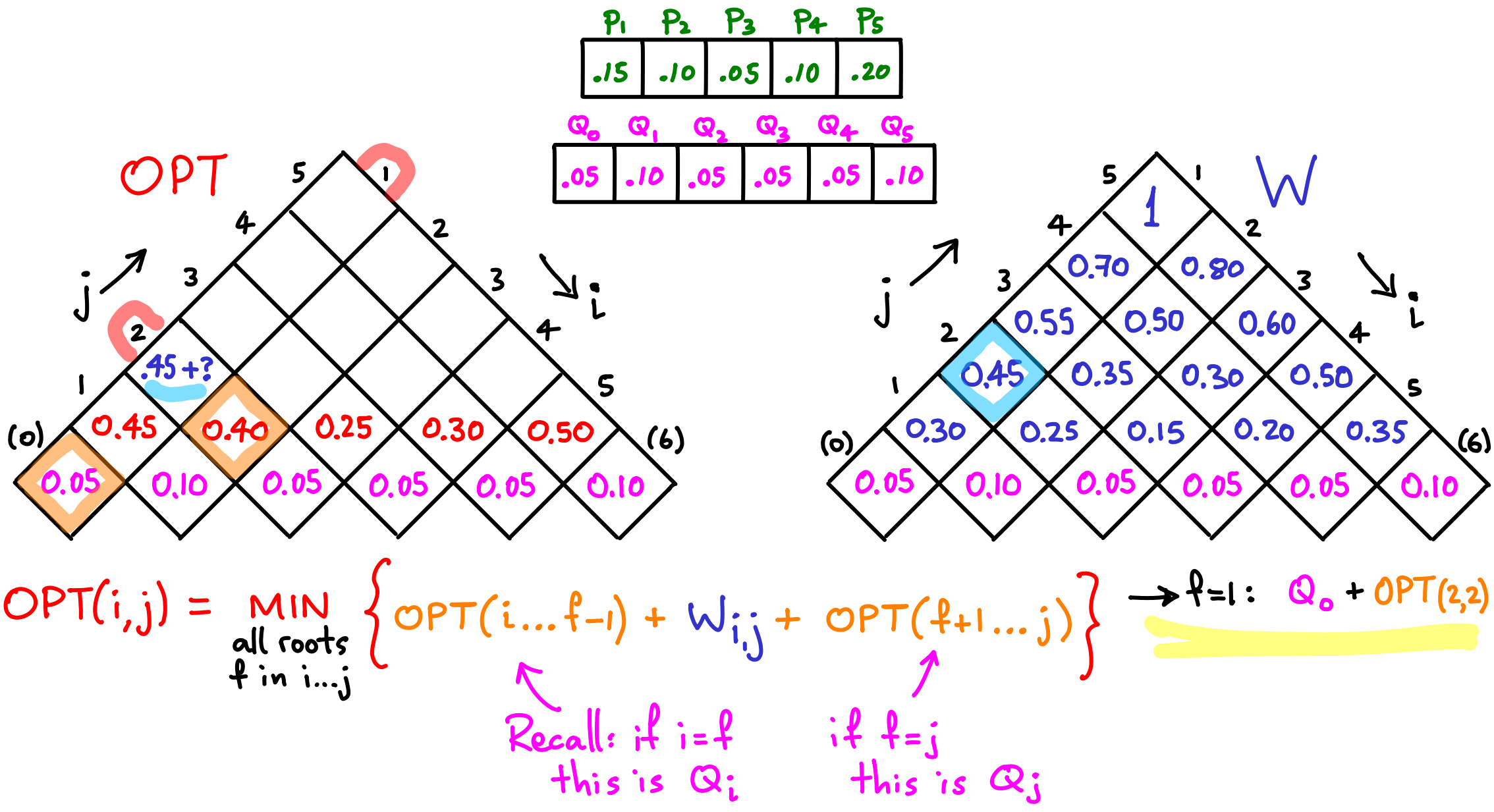
if $f=j$ this is Q_j

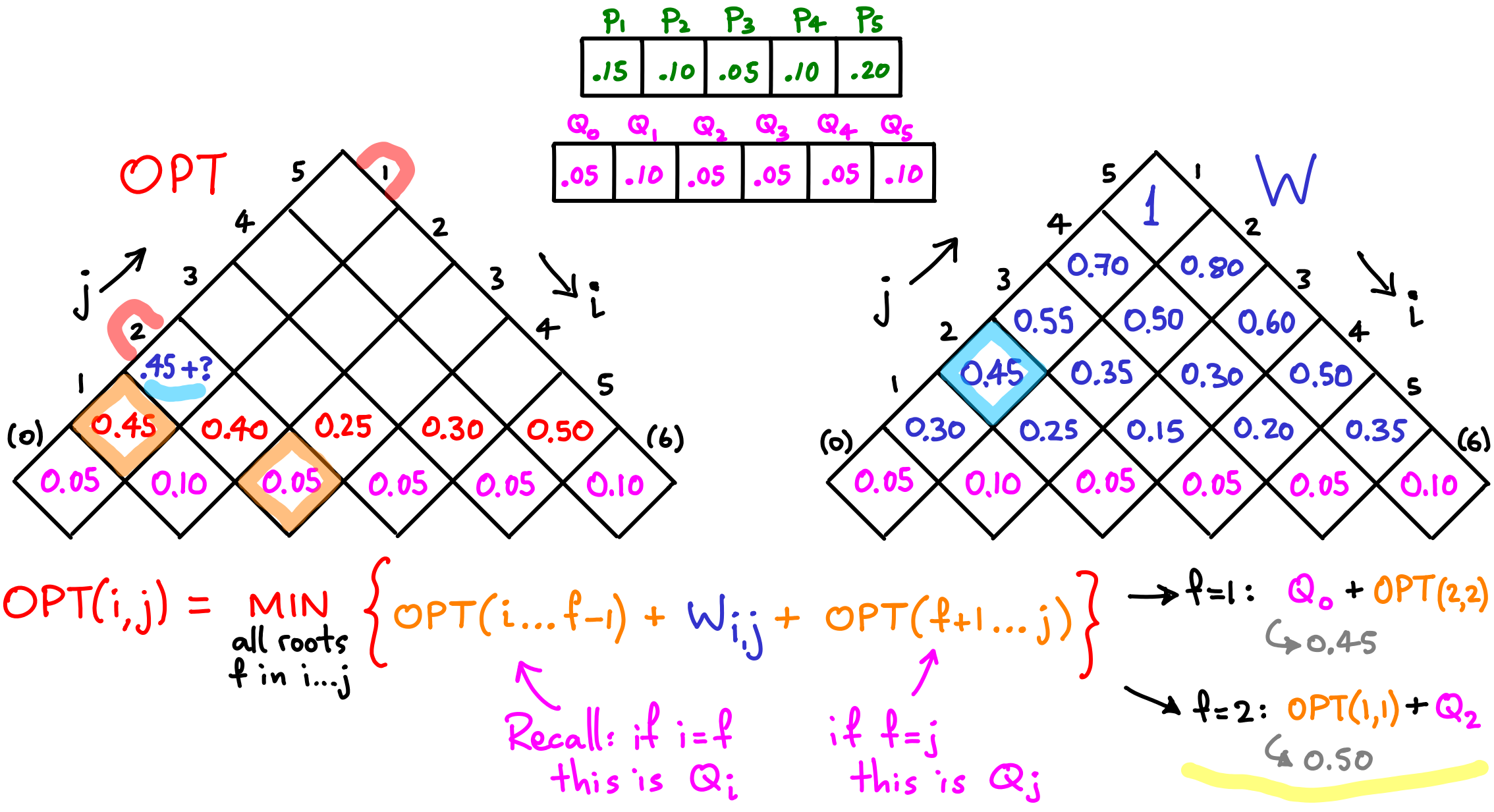


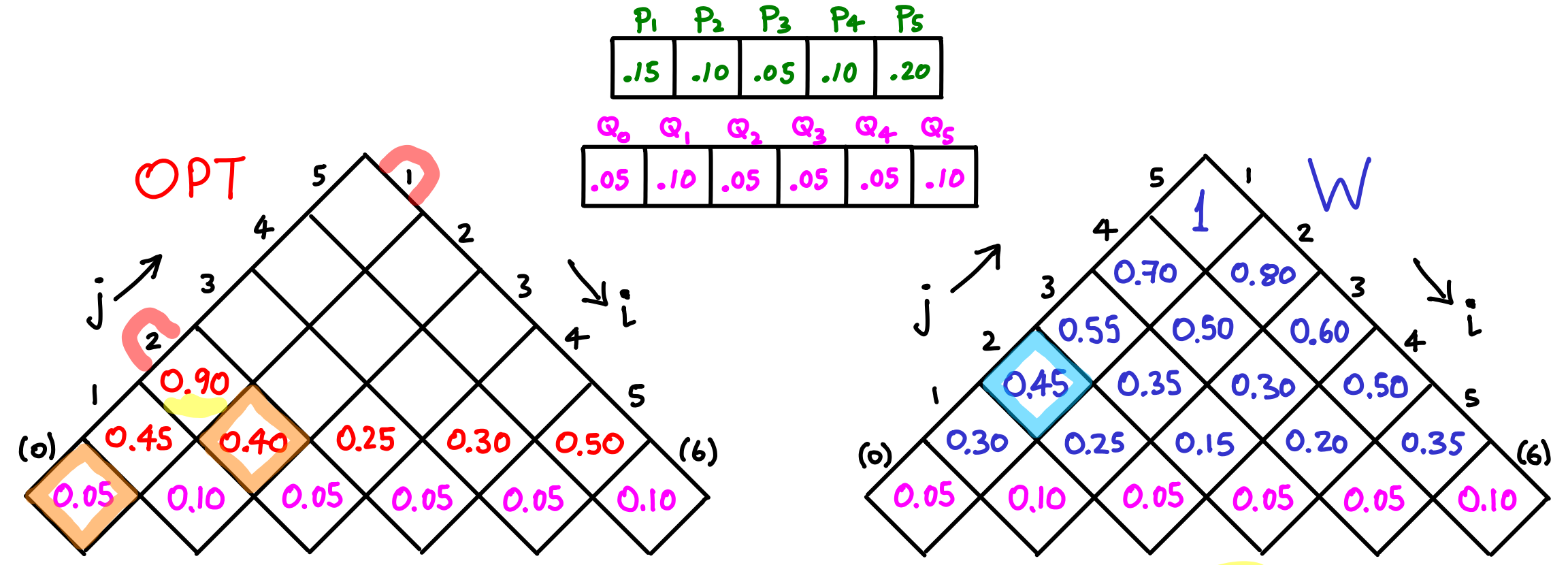
$$OPT(i,j) = \min_{\text{all roots } f \text{ in } i\ldots j} \{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \}$$

Recall: if $i=f$ this is Q_i

if $f=j$ this is Q_j


P1 P2 P3 P4 P5
.15 .10 .05 .10 .20
Q0 Q1 Q2 Q3 Q4 Q5
.05 .10 .05 .05 .05 .10
OPT
W
i
j
(0)
(6)
.45 +?
0.45 0.40 0.25 0.30 0.50
0.05 0.10 0.05 0.05 0.05 0.10
1
0.70 0.80
0.55 0.50 0.60
0.45 0.35 0.30 0.50
0.30 0.25 0.15 0.20 0.35
0.05 0.10 0.05 0.05 0.05 0.10
OPT(i,j) = MIN all roots f in i...j { OPT(i...f-1) + W_ij + OPT(f+1...j) }
→ f=1: Q0 + OPT(2,2)
Recall: if i=f this is Qi
if f=j this is Qj


P1 P2 P3 P4 P5
.15 .10 .05 .10 .20
Q0 Q1 Q2 Q3 Q4 Q5
.05 .10 .05 .05 .05 .10
OPT
W
i
j
1
2
3
4
5
(0)
(6)
.45+?
0.45 0.40 0.25 0.30 0.50
0.05 0.10 0.05 0.05 0.05 0.10
0.70 0.80
0.55 0.50 0.60
0.45 0.35 0.30 0.50
0.30 0.25 0.15 0.20 0.35
OPT(i,j) = MIN all roots f in i...j { OPT(i...f-1) + W_ij + OPT(f+1...j) }
Recall: if i=f this is Q_i
if f=j this is Q_j
f=1: Q0 + OPT(2,2)
0.45
f=2: OPT(1,1) + Q2
0.50



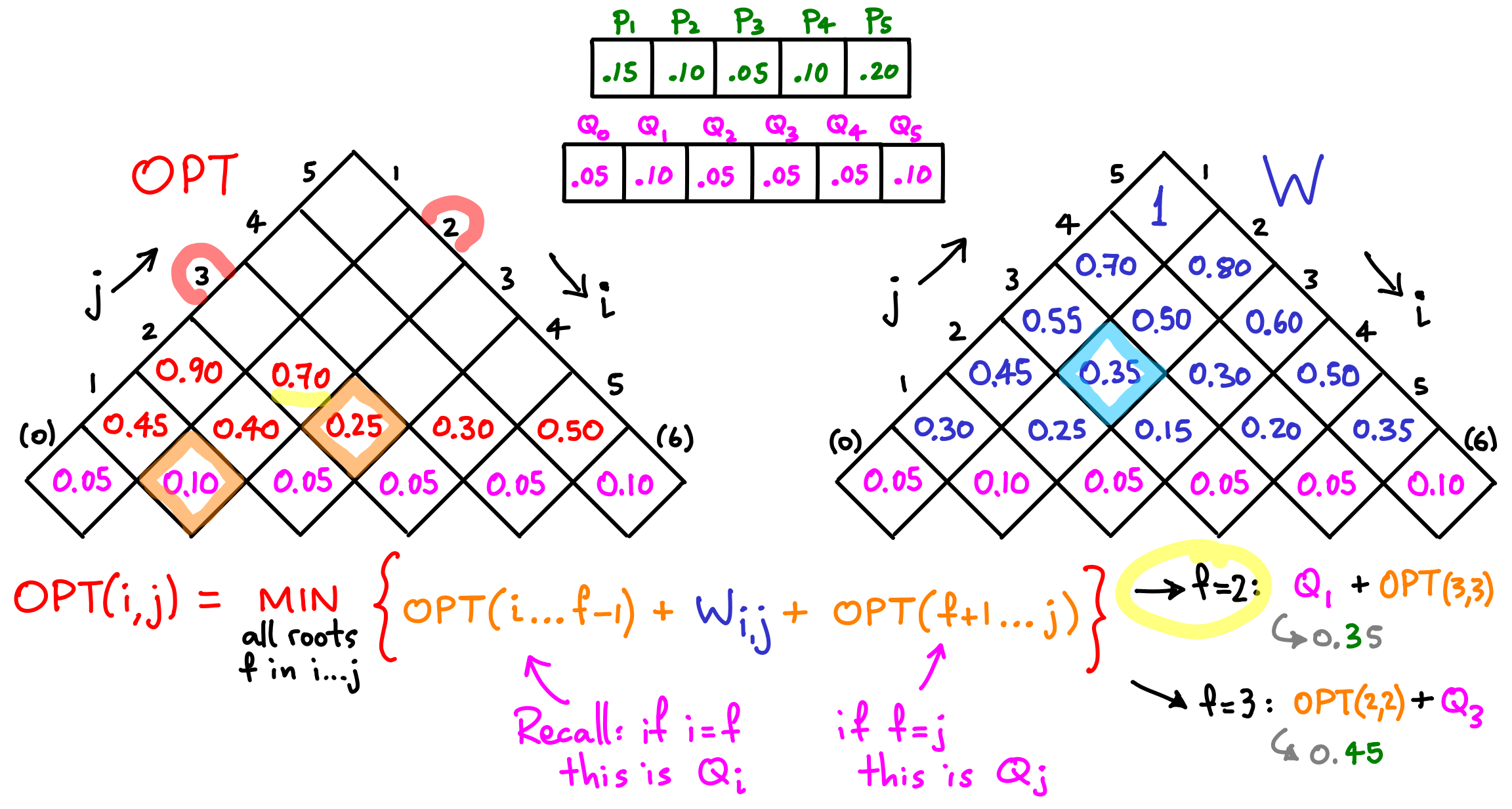
$$OPT(i,j) = \underset{\text{all roots } f \text{ in } i\ldots j}{\text{MIN}} \left\{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \right\}$$

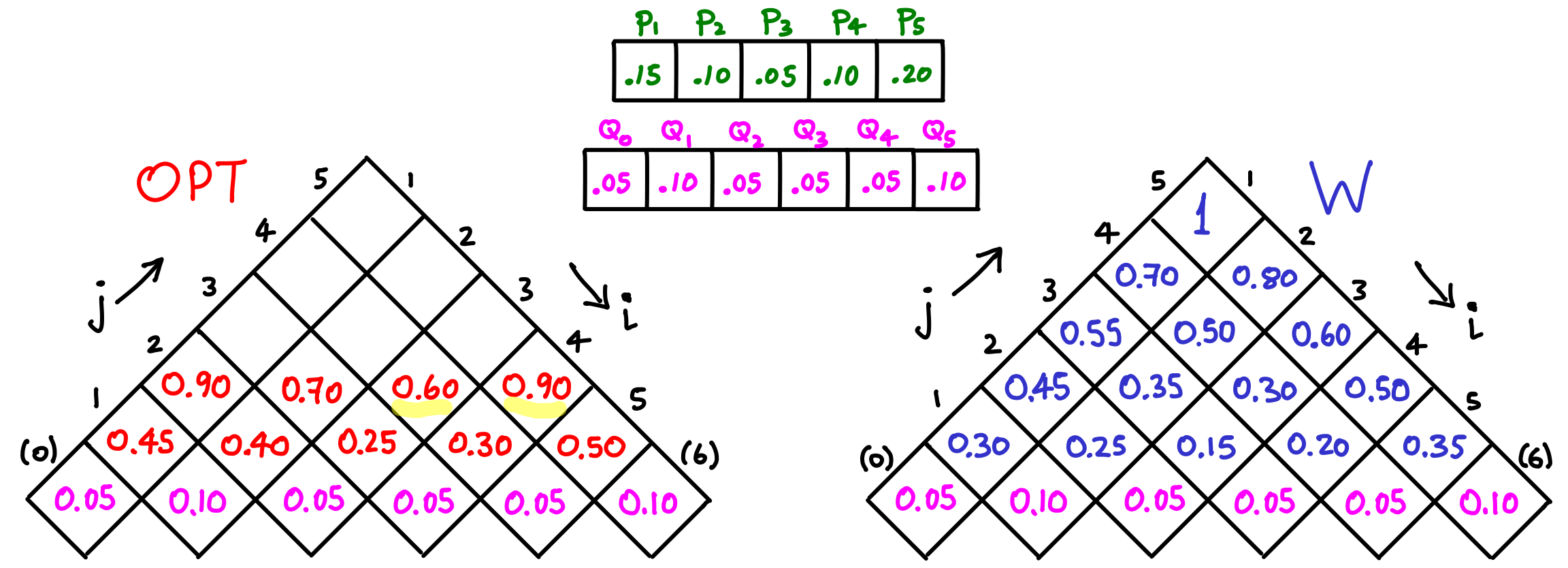
Recall: if $i=f$ this is Q_i

if $f=j$ this is Q_j

→ $f=1$: $Q_0 + OPT(2,2)$ ↳ 0.45

→ $f=2$: $OPT(1,1) + Q_2$ ↳ 0.50

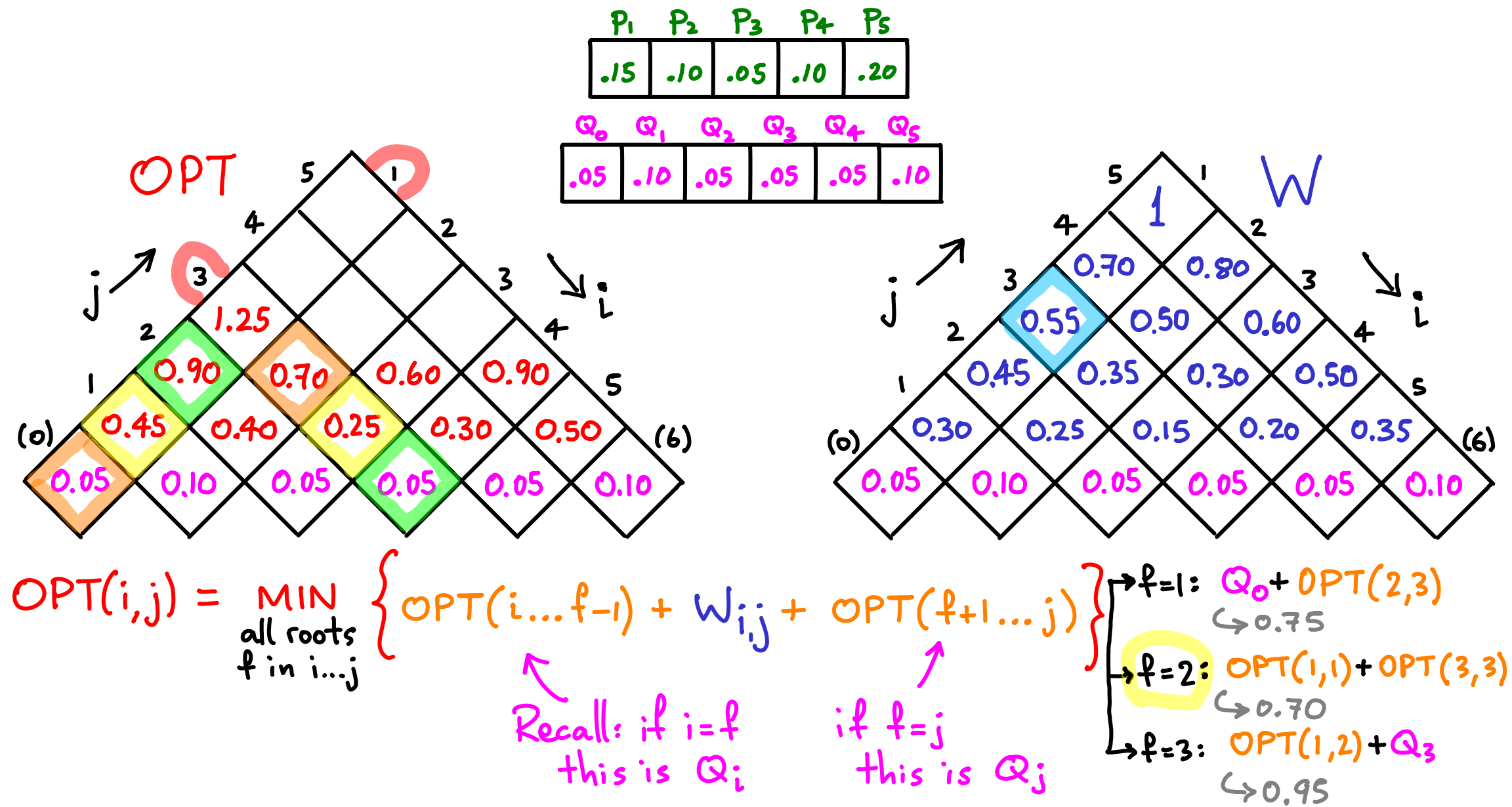

P1 P2 P3 P4 P5
.15 .10 .05 .10 .20
Q0 Q1 Q2 Q3 Q4 Q5
.05 .10 .05 .05 .05 .10
OPT
j
i
(0)
(6)
0.90
0.70
0.45
0.40
0.25
0.30
0.50
0.05
0.10
0.05
0.05
0.05
0.10
W
1
0.70
0.80
0.55
0.50
0.60
0.45
0.35
0.30
0.50
0.30
0.25
0.15
0.20
0.35
0.05
0.10
0.05
0.05
0.05
0.10
OPT(i,j) = MIN all roots f in i...j { OPT(i...f-1) + W_ij + OPT(f+1...j) }
Recall: if i=f this is Q_i
if f=j this is Q_j
f=2: Q1 + OPT(3,3)
0.35
f=3: OPT(2,2) + Q3
0.45

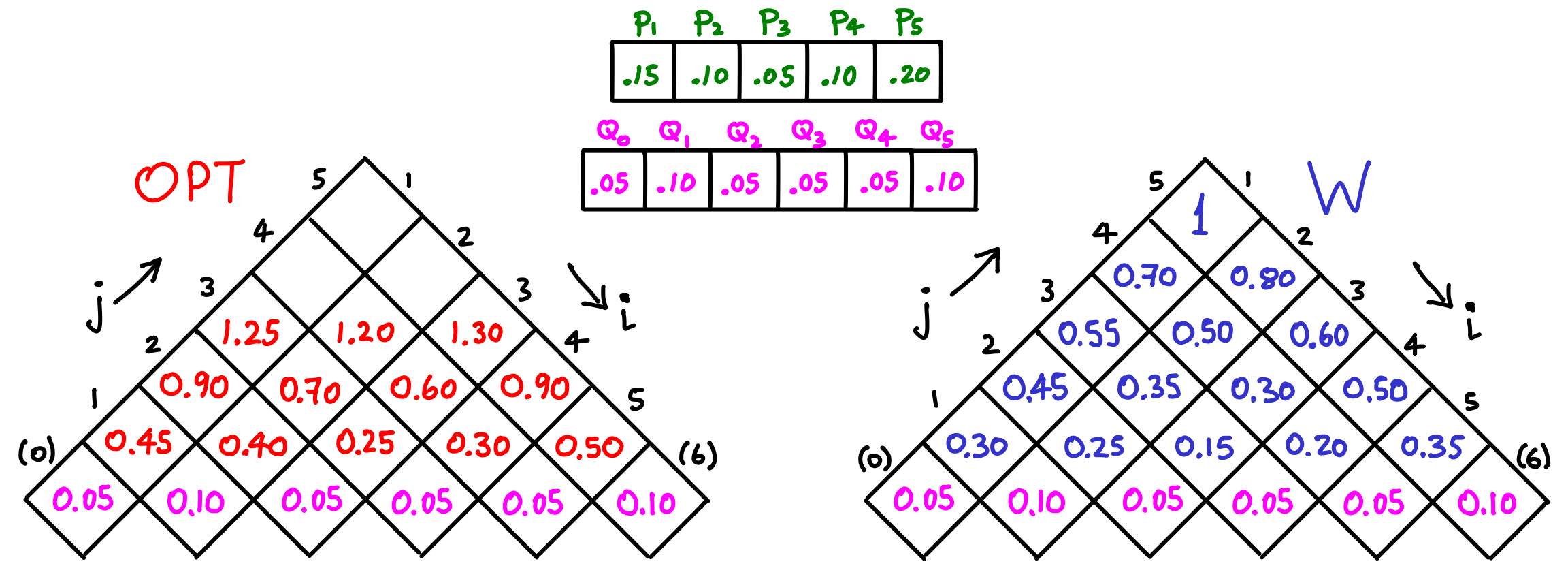


$$OPT(i,j) = \underset{\text{all roots } f \text{ in } i\ldots j}{\text{MIN}} \left\{ OPT(i \ldots f-1) + W_{ij} + OPT(f+1 \ldots j) \right\}$$

Recall: if $i=f$ this is Q_i

if $f=j$ this is Q_j

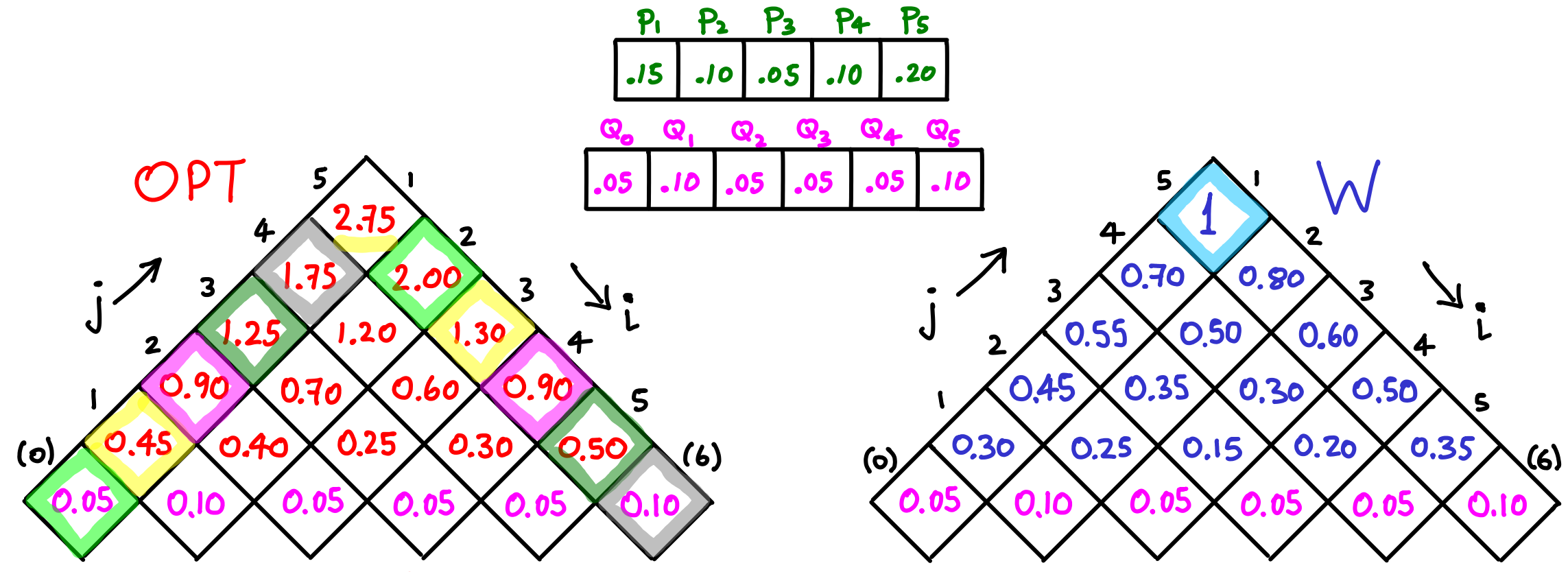

P1 P2 P3 P4 P5
.15 .10 .05 .10 .20
Q0 Q1 Q2 Q3 Q4 Q5
.05 .10 .05 .05 .05 .10
OPT
j
i
1.25
0.90 0.70 0.60 0.90
0.45 0.40 0.25 0.30 0.50
0.05 0.10 0.05 0.05 0.05 0.10
(0)
(6)
W
0.70 0.80
0.55 0.50 0.60
0.45 0.35 0.30 0.50
0.30 0.25 0.15 0.20 0.35
0.05 0.10 0.05 0.05 0.05 0.10
OPT(i,j) = MIN all roots f in i...j { OPT(i...f-1) + W_ij + OPT(f+1...j) }
Recall: if i=f this is Q_i
if f=j this is Q_j
f=1: Q0 + OPT(2,3) → 0.75
f=2: OPT(1,1) + OPT(3,3) → 0.70
f=3: OPT(1,2) + Q3 → 0.95



$$OPT(i,j) = \min_{\text{all roots } f \text{ in } i\ldots j} \{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \}$$

Recall: if $i=f$ this is Q_i

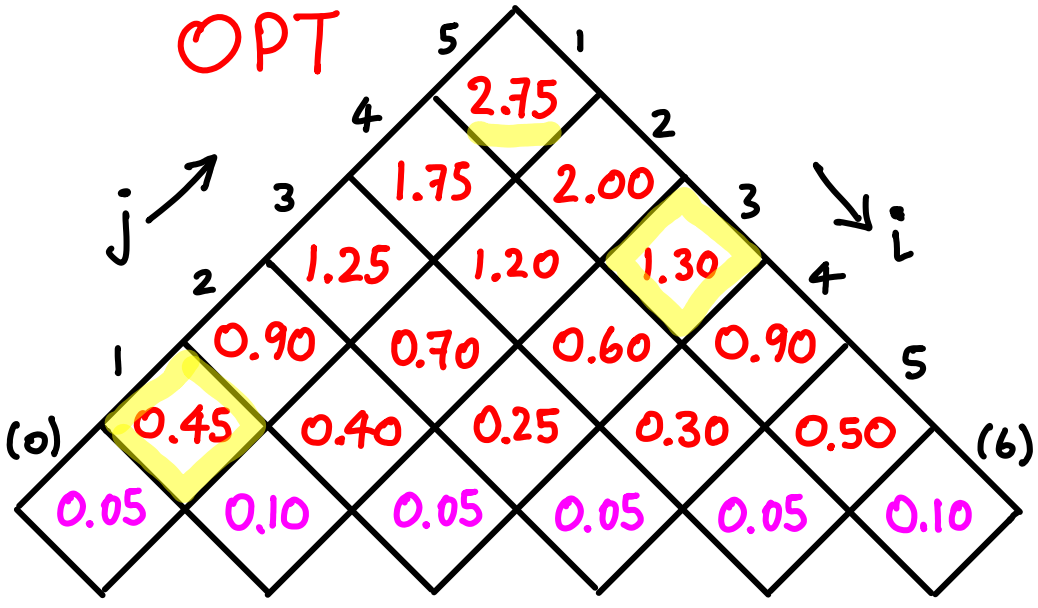
if $f=j$ this is Q_j

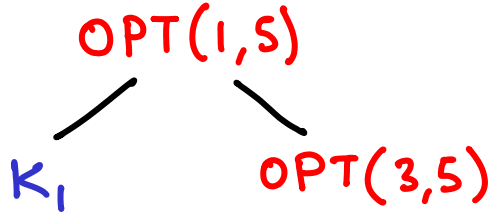


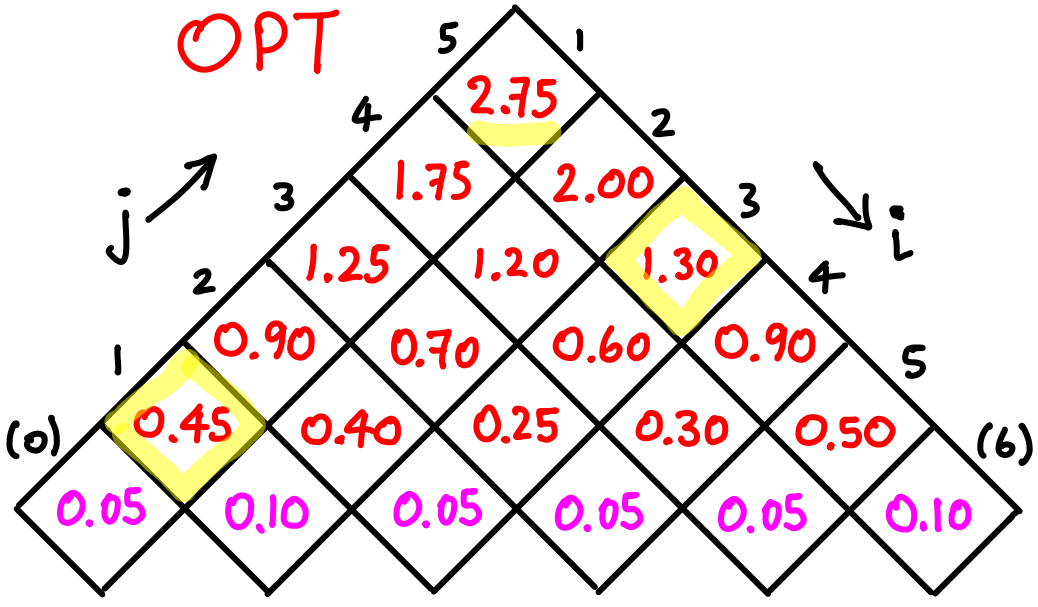
$$OPT(i,j) = \min_{\text{all roots } f \text{ in } i \ldots j} \{ OPT(i \ldots f-1) + W_{i,j} + OPT(f+1 \ldots j) \}$$

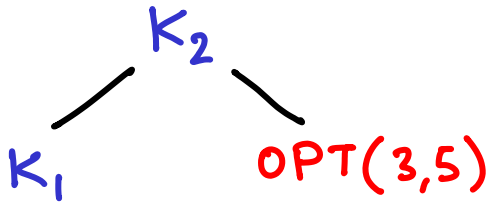
Recall: if $i=f$ this is Q_i

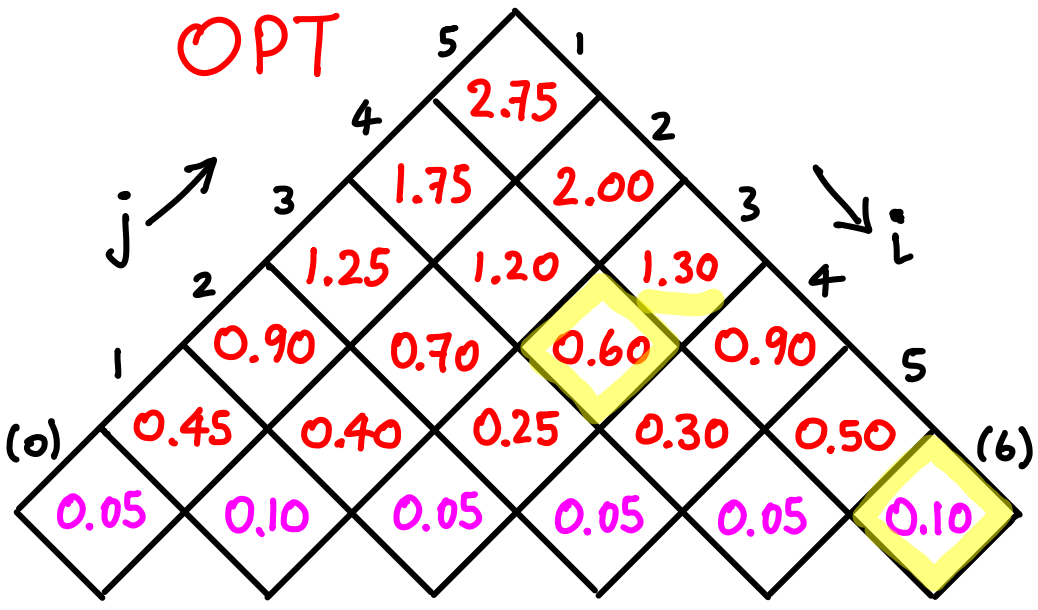
if $f=j$ this is Q_j

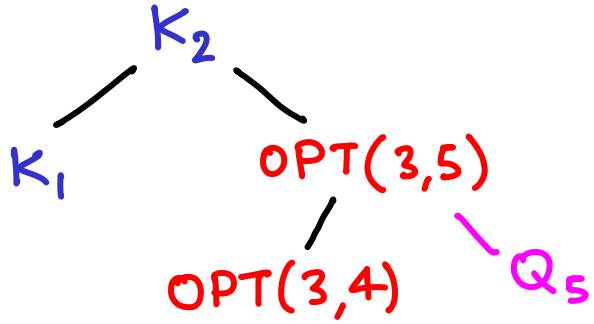

OPT
5
1
2.75
4
2
1.75
2.00
j
3
3
i
1.25
1.20
1.30
2
4
0.90
0.70
0.60
0.90
1
5
(0)
0.45
0.40
0.25
0.30
0.50
(6)
0.05
0.10
0.05
0.05
0.05
0.10

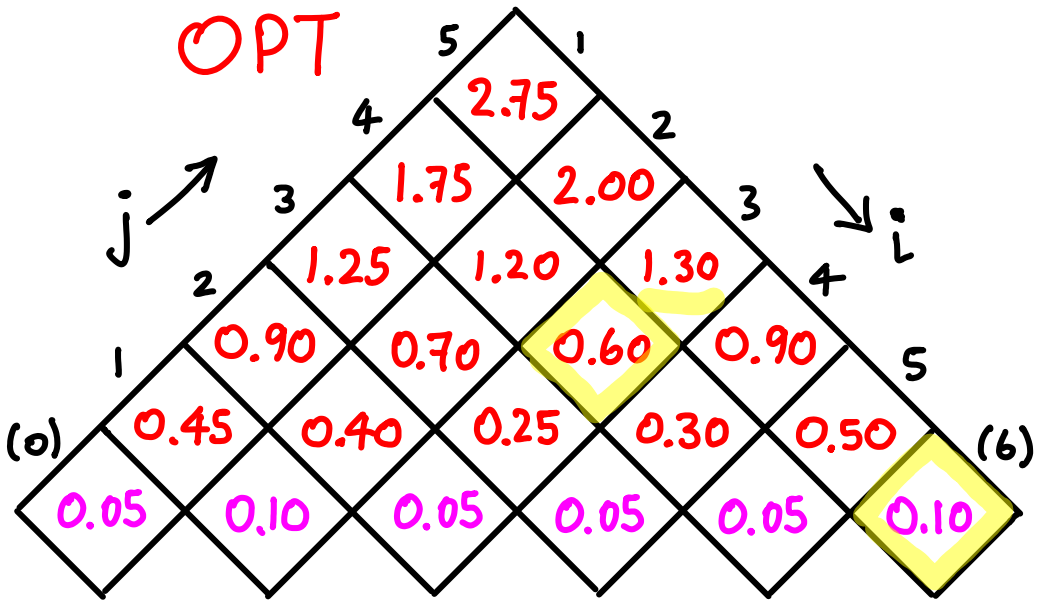

OPT(1,5)
K_1
OPT(3,5)

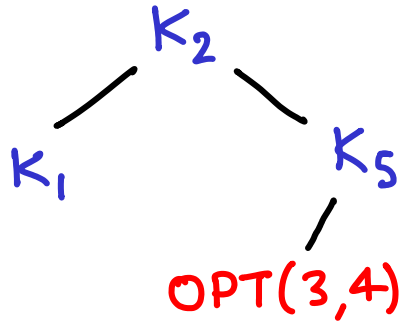

OPT
j
i
5
1
4
2
3
3
2
4
1
5
(0)
(6)
2.75
1.75
2.00
1.25
1.20
1.30
0.90
0.70
0.60
0.90
0.45
0.40
0.25
0.30
0.50
0.05
0.10
0.05
0.05
0.05
0.10

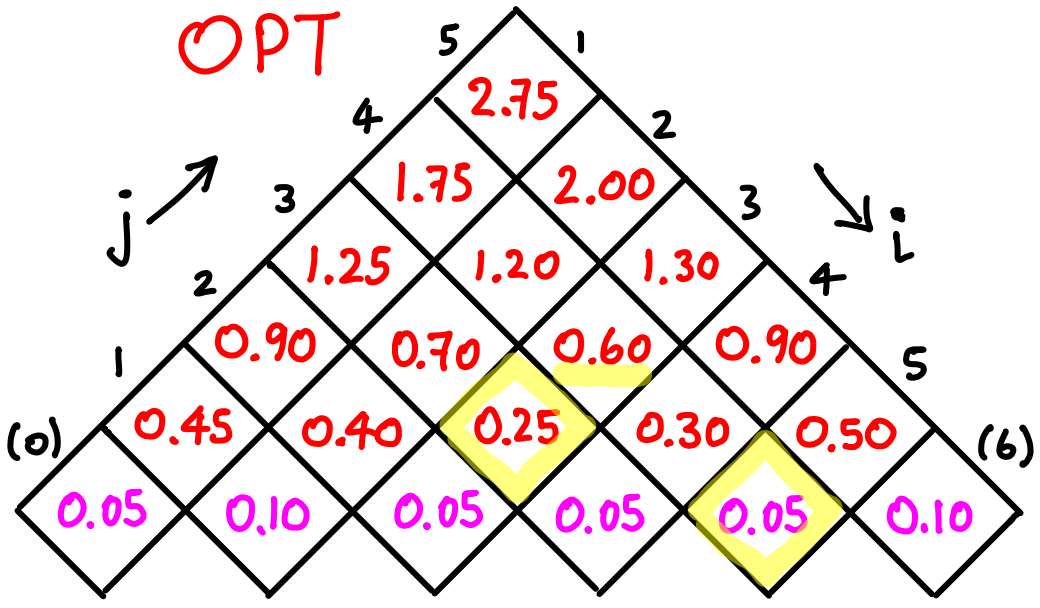

K2
K1
OPT(3,5)

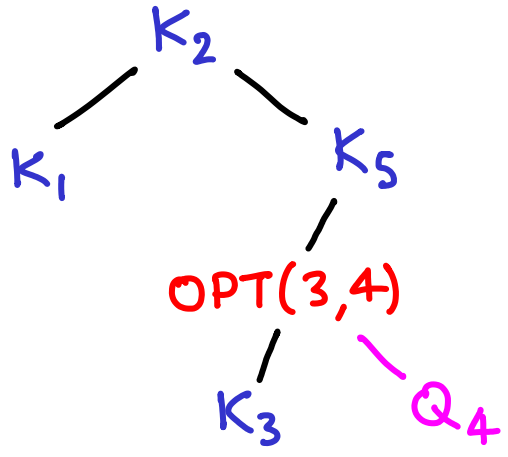

OPT
j
i
(0)
(6)
5
1
2.75
4
2
1.75
2.00
3
3
1.25
1.20
1.30
2
4
0.90
0.70
0.60
0.90
1
5
0.45
0.40
0.25
0.30
0.50
0.05
0.10
0.05
0.05
0.05
0.10

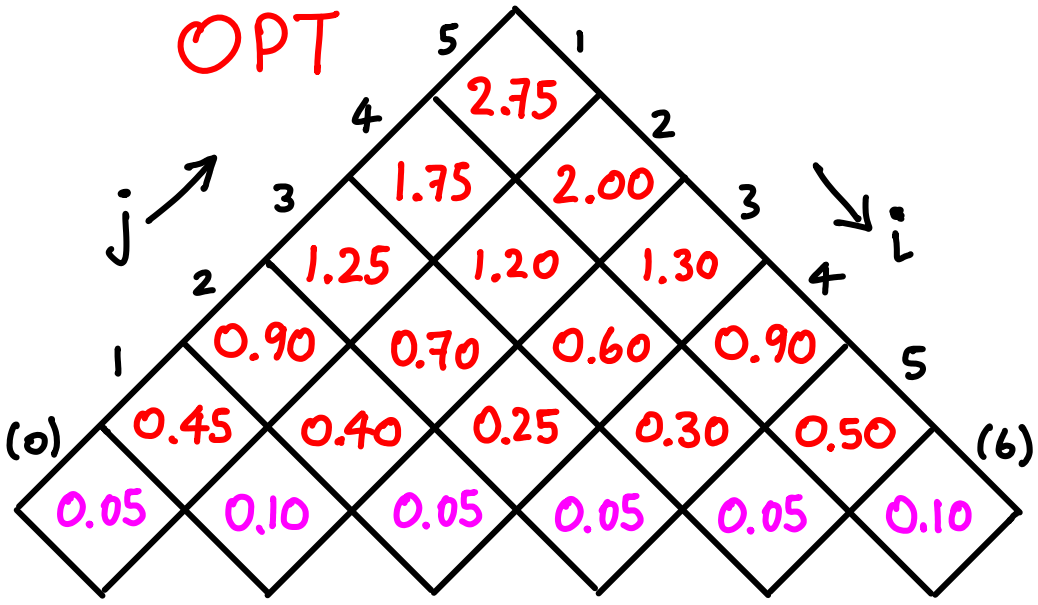

K2
K1
OPT(3,5)
OPT(3,4)
Q5

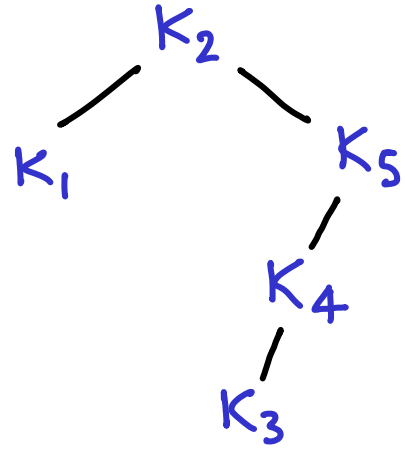

OPT
j
i
5
1
4
2
3
3
2
4
1
5
(0)
(6)
2.75
1.75
2.00
1.25
1.20
1.30
0.90
0.70
0.60
0.90
0.45
0.40
0.25
0.30
0.50
0.05
0.10
0.05
0.05
0.05
0.10


K2
K1
K5
OPT(3,4)


OPT
j
i
(0)
(6)
5
1
4
2
3
3
2
4
1
5
2.75
1.75
2.00
1.25
1.20
1.30
0.90
0.70
0.60
0.90
0.45
0.40
0.25
0.30
0.50
0.05
0.10
0.05
0.05
0.05
0.10

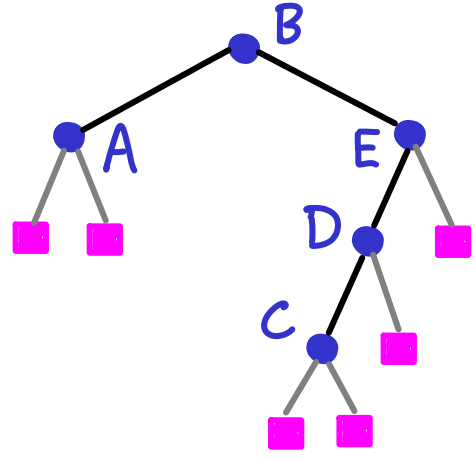

K2
K1
K5
OPT(3,4)
K3
Q4





P_1	P_2	P_3	P_4	P_5
.15	.10	.05	.10	.20

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
.05	.10	.05	.05	.05	.10



$$1 \cdot .10 + 2 \cdot (.15 + .20) + 3 \cdot (.05 + .10 + .10 + .10) + 4 \cdot (.05 + .05) + 5 \cdot (.05 + .05)$$