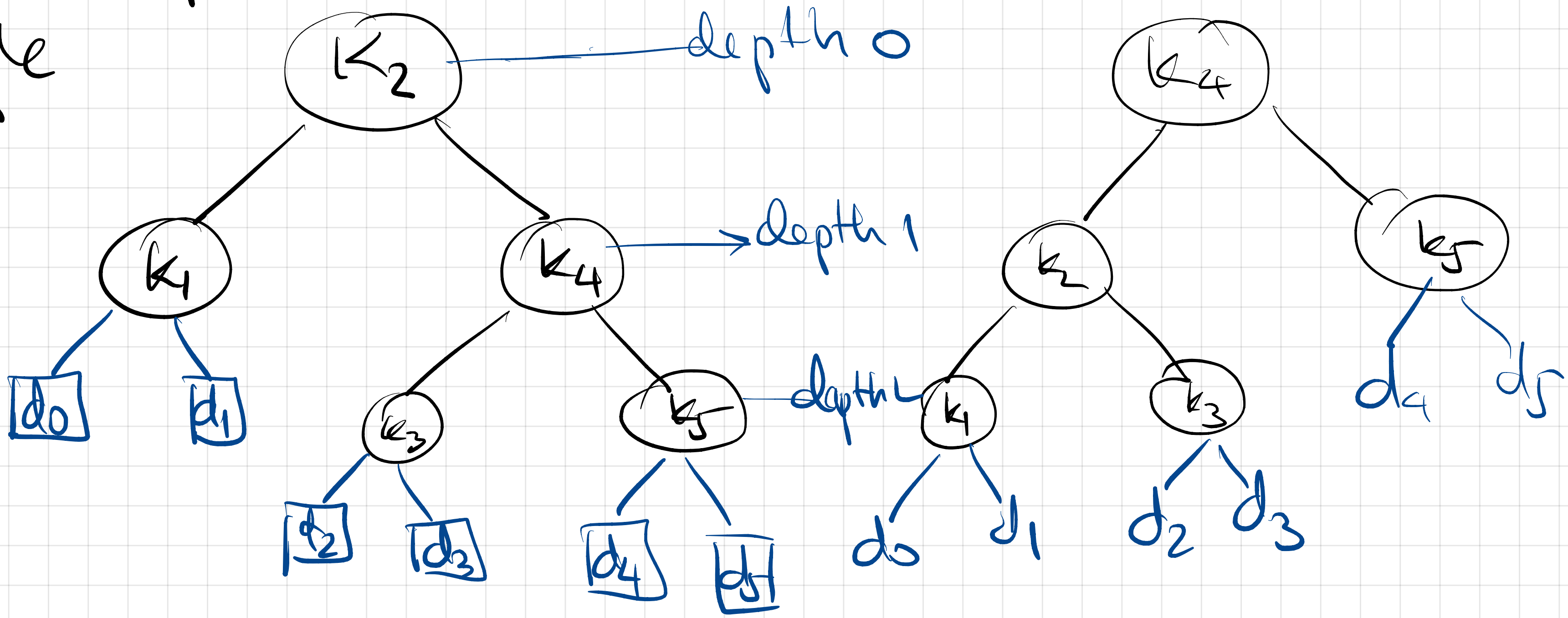
LECTURE 8 Optimal BST

ordered values $k_1 \le k_2 \le \dots \le$

example BST



Search probability : $Pr(k_i) = p_i$ not uniform

$Pr(d_i) = q_i$

$d_i =$ searches for values $k_i < val < k_{i+1}$

$$\sum p_i + \sum q_i = 1$$

depth = level

OPTIMALITY : min expected search cost

$$\sum_{i=1}^{n} [\text{depth}(k_i) + 1] \, p_i + \sum_{i=0}^{n} [\text{depth}(d_i) + 1] \cdot q_i$$

Step 1 OPTSOL given

$k_1 < \ldots <$ k_r $< \ldots k_n$

Left subpb

Right subpb.

PbL 0.6

sum to $1-p_r$ not 1

0.4 PbR

$\sum p_{1:r-1}$ all probs for
$+ \sum q_{0:r-1}$ Left side

$k[1:r-1]$

$1 \le k \le r-1$

$d[0:r-1]$

$k[r+1:n]$

$d[r:n]$

$p_1, \ldots p_{r-1}$
$q_0 \ldots q_{r-1}$

example $0.6 = $ PbL

PbR 0.4 $1-$PbL

$p_{r+1} \ldots p_n$
$q_r \ldots q_n$

Search: $0.6 \cdot 1^{root}$ + best way to search $(k[1:r-1])$

Search: $0.4 \cdot 1^{root}$ + best way to search $k[r+1:n]$

② $C[i,j]$ = best cost for keys $k_i, k_{i+1}, k_r \ldots k_j$ tree (as part of overall cost)

$T[i,j]$ = height (# levels) of the best tree for keys$[i:j]$

$$W[i,j] = \text{total prob } [i:j] = \sum_{t=i}^{j} p_t + \sum_{t=i-1}^{j} q_t$$

$C[i,j]$ = best expected cost in best$[i,j]$ subtree =

= height(subtree$[i:j]$) x prob(search_subtree$[i,j]$) = $T[i,j] \times W[i,j]$

divide $C[i,j]$ into 3 disjoint searches:

= cost search (k_r) + cost search leftsubtree $[i:r-1]$ + cost search rightsubtree $[r+1:j]$

$$= \underbrace{1}_{\text{step}} \cdot \underbrace{p_r}_{\text{prob}} + \underbrace{W[i,r-1]}_{\text{prob(left)}} \cdot (\underbrace{1}_{\text{root 1 step}} + \text{height leftsubtree})_{\text{\# steps}} + \underbrace{W[r+1:j]}_{\text{prob\_right}} \cdot \underbrace{(1 + \text{height\_right\_sub})}_{\text{\# steps}}$$

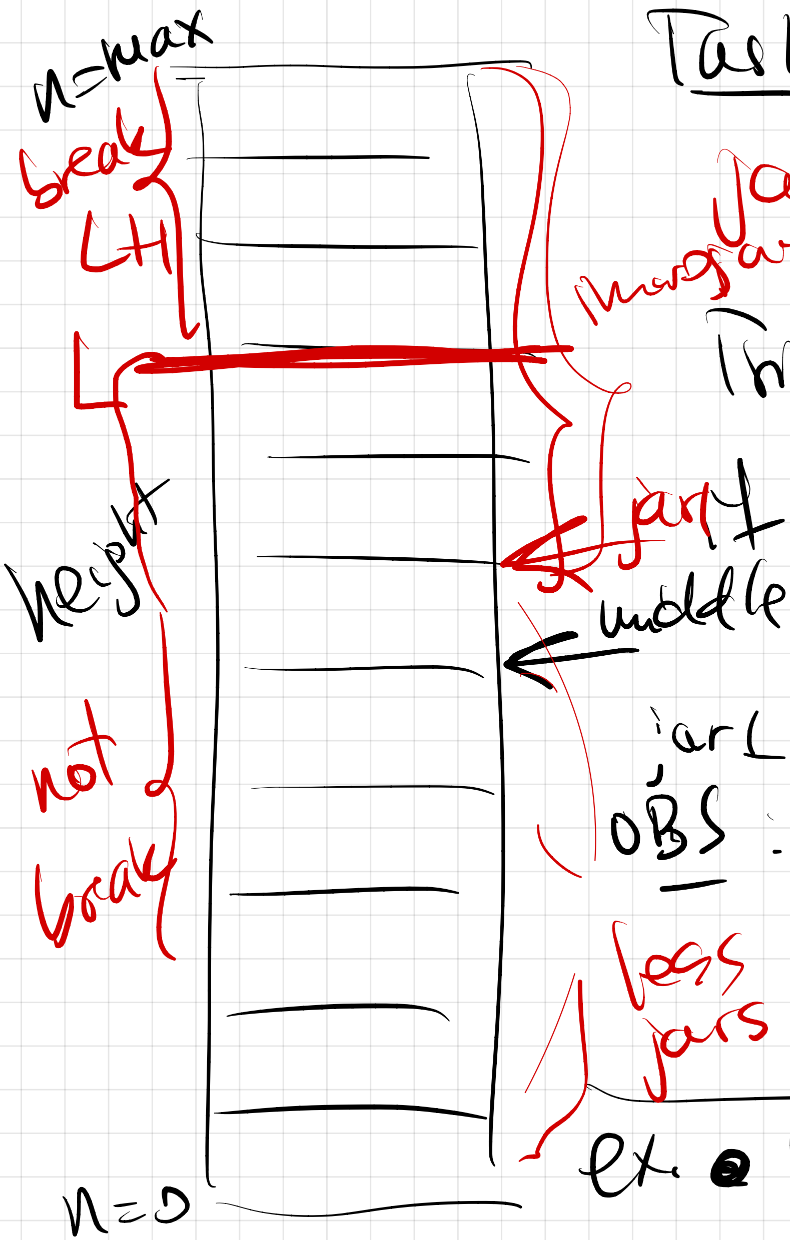
$$= 1 \cdot p_r + W[i,r-1](1 + T[i,r-1]) + W[r+1:j](1 + T[r+1,j])$$

$$= \underbrace{p_r + w[i,r-1] + w[r+1:j]}_{W[i,j]} + \underbrace{W[i,r-1] \cdot T[i,r-1]}_{C[i,r-1]} + \underbrace{W[r+1,j] \cdot T[r+1,j]}_{C[r+1,j]}$$

$$= W[i,j] + C[i,r-1] + C[r+1,j]$$

Jars on Ladder n steps k jars

Task: find highest Level L where jars don't break (they break at $L+1$)



Trial jar at level ℓ

$\ell \leq L \Rightarrow$ doesn't break $\Rightarrow$ can reuse it

$\ell \geq L+1 \Rightarrow$ jar breaks cannot reuse it

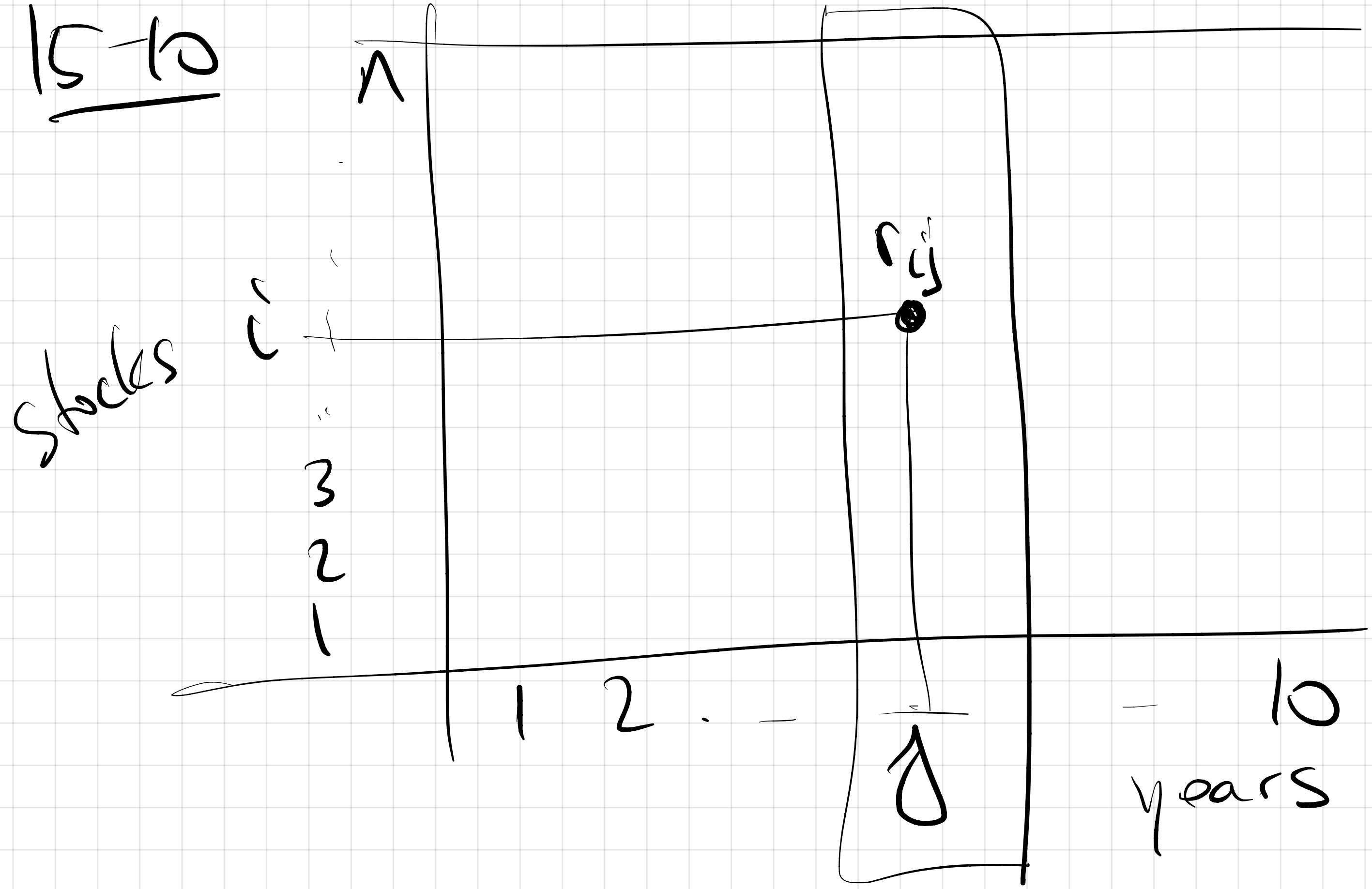
OBS: minimum # trials in worst case

You must 100% find L

ex. • $k=1 \Rightarrow$ bottom up on stair $\Theta(n)$

• $k > \log n \Rightarrow$ binary search

15-10



r_{ij} = return % of stock i year j

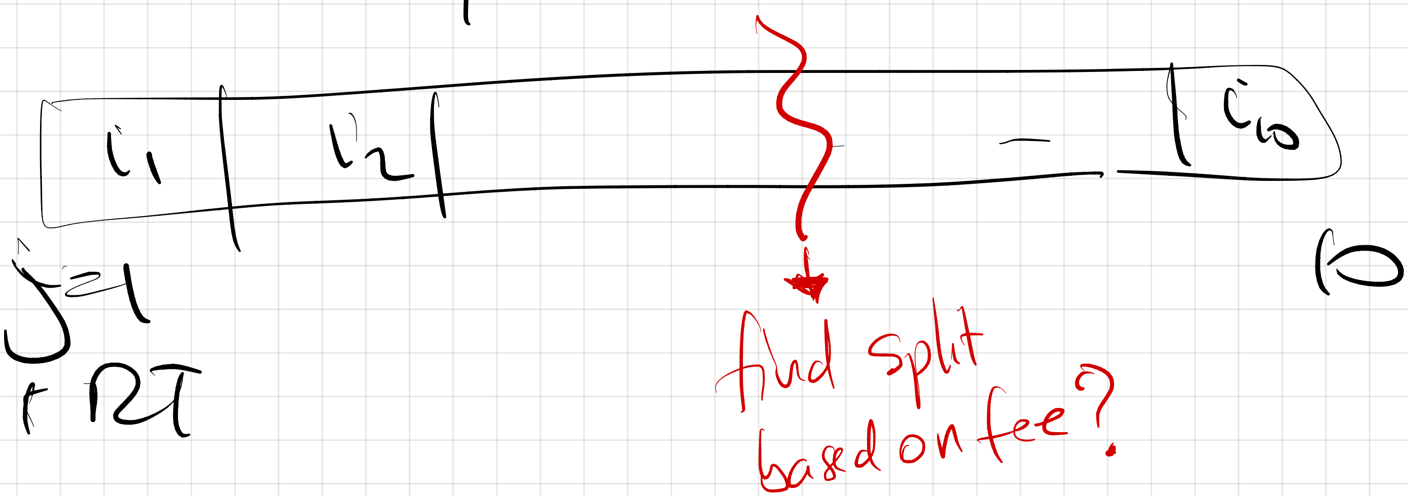
put

$\$N \to \$N + \$N\, r_{ij}$

$\$N \to (1+r_{ij})\$N$

a) pick best stock that year

b) OPT SOL



c) pseudocode + RT

d) max allocation per stock \$15000

OPTSOL charact ?

prev

NOT WORKING

still DP ?

Speculation : – DP structure works

– C[] table of subph

has to be bounded ?

Painters (fence problem)   fence = boards sequence

b_i = length of board i

K painters   same speed   paint done / unit

painter 1: $b_1 \; \dots \; b_{p_2}$ → time 1 = $\sum$ length(boards) $= \sum_{i=1}^{p_2} b_i$ (first painter)

2: $b_{p_2+1} \; \dots \; b_{p_3}$

K−1: $b_{p_{k-1}+1} \; \dots \; b_{p_k}$

K: $b_{p_k+1} \; \dots \; b_n$

Task: partition boards (find $p_2, p_3, \dots p_k$)
such that the longest job is minimum
(scenario: painters work in parallel ⇒
⇒ time to finish ≃ longest job/partition)

$C[j, k]$ = best (min time in parallel) to paint boards $1:j$ with painters $1, 2, \dots, k$

$1 \le j \le n$

j → board index

k → #painters

$$= \underset{k-1 \le p_k \le n-1}{\text{MIN search for}} \ \underline{\underline{\text{MAX}}} \left\{ C[1:p_k, k-1] \ ; \ \text{last job } \sum_{t=p_k+1}^{n} b_t \right\}$$

??

p_k = last board painted by painter $k-1$

$\Theta(n)$

Total R.T. $\Theta(nk \times n)$

search in $\Theta(\log n)$ time $\longrightarrow \Theta(n\, k \log n)$

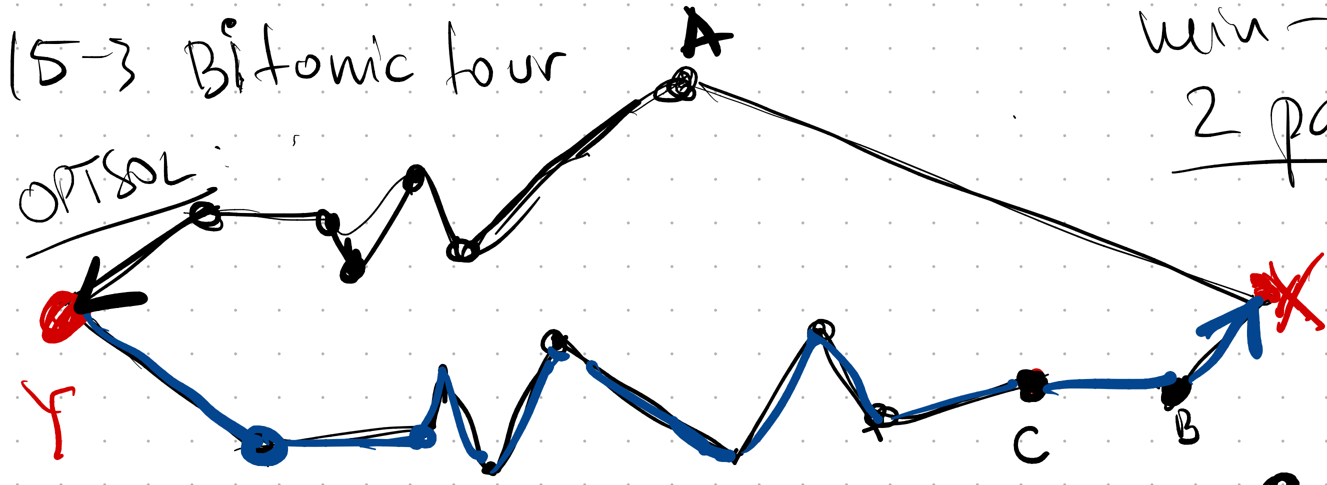
exercise

$$\underset{\text{candidates } p_k}{\Delta} ? = C[p_k, k-1] - \left(\text{last job } \sum_{t=p_k+1}^{n} b_t \right)$$

$$\sum_{t=p_k+1}^{n} b_t = \underbrace{B_n}_{\substack{\text{cumul} \\ \sum_{t=1}^{n} b_t}} - \underbrace{B_{p_k}}_{\substack{\text{cumul} \\ \sum_{t=1}^{p_k} b_t}}$$

NEXT: Selected previous midterm questions

15-3 Bitonic tour

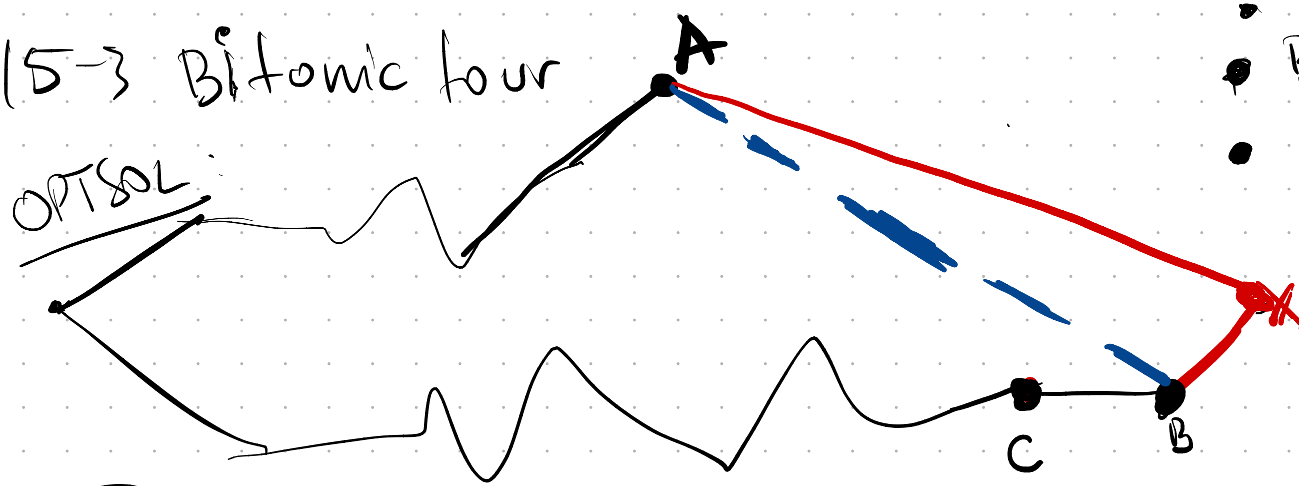


min-distance-total tour

2 paths : X, Y extreme L, R

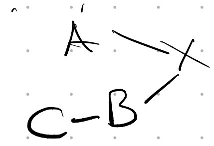
- X → Y upper path
- Y → X lower path
- each point ~~in~~ one of the paths.
- each path strictly directional L→R (lower) or R→L upper.

15-3 Bitonic tour



- A, B connect to X
- B closest (right most) to X
- C connects to B

$\Longrightarrow$



SPB = PB \ {x} right most B

Idea! OPTSOL \ {x} is optimal for PB \ {x} = SPB ? Maybe Yes

assume (contradict hypothesis) there is a better solution S for SPB = PB \ {x}

Then solution is:

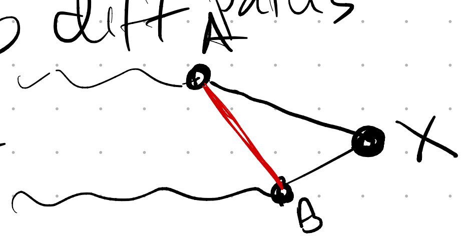
- eliminate X, solve SPB = PB \ {x} with B right-most
- then connect X either to AB or to BC, whichever is better

$$C[PB_X] = C[SPB_B] + \text{best}\{+XA+XB-AB,\ +XB+XC-BC\}$$

idea 2

A, B ≥ X closest to X ⇒ 2 possibilities in OPTSOL (A ≥ B ≥ X closest)

- A, B diff paths



OPTSOL

$SPB = PB \setminus \{x\}$

$AB \in sol(SPB)$

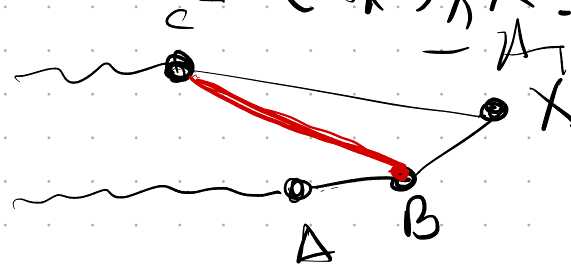
$OPTSOL \setminus \{AX, BX\} = sol(SPB) \setminus \{AB\}$

or exchange argument

Solution =

$= sol(SPB) + \underbrace{XA + XB - AB}_{\text{connect } X}$

- A, B same path

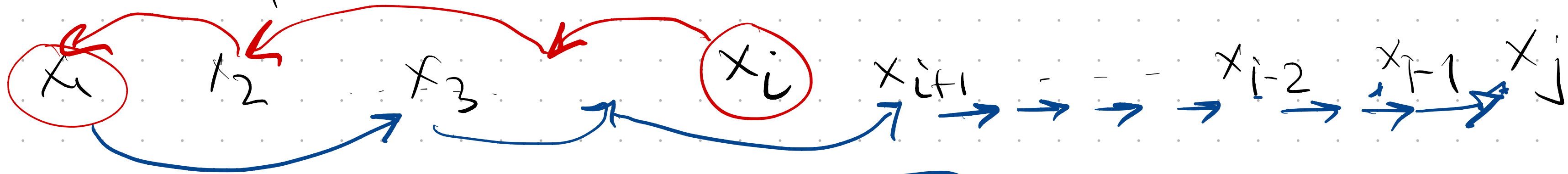


$SPB = PB \setminus \{B\}$ where X is still right-most but A is closest

$SOLUTION = sol(SPB) + \underbrace{BA + BX - XA}_{\text{connect } B}$

idea3

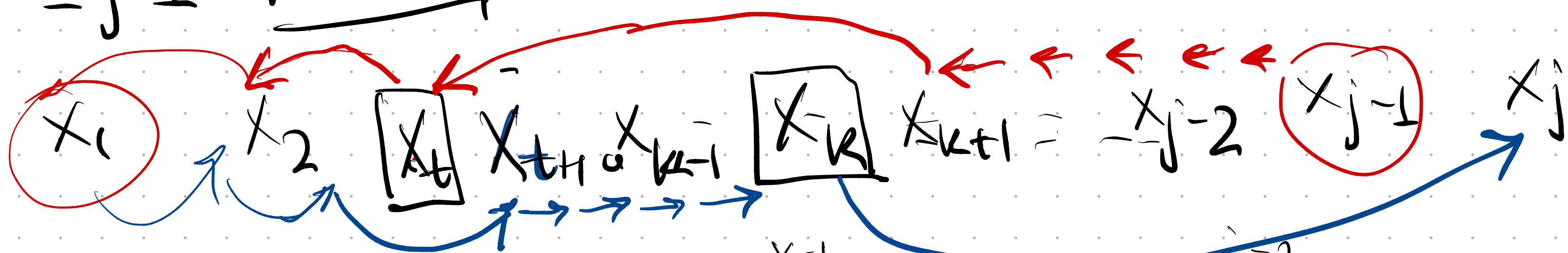
Sort x-positions 1–j. Pick x_i.



want bitonic tour $x_i \rightsquigarrow x_1$ Left $\rightsquigarrow x_j$ Right

$c[i,j]$ = best bitonic path objective from x_i to x_j passing through all points $x_1, x_2, \dots x_i, \dots x_j$ sorted.

- if $i \quad j-1$ easy: $c[i,j] = c[i,j-1] + dist(x_{j-1}, x_j)$
- if $i = j-1$ ~~not easy~~: search for x_k that right-jumps $x_k \curvearrowright x_j$



$$c[j-1,j] = \text{best}_{t,k} \left(\sum_{\ell=t}^{k-1} \|x_\ell x_{\ell+1}\| + \sum_{\ell=k+1}^{j-2} \|x_\ell x_{\ell+1}\| + c[t,k] + \|x_t x_{k+1}\| + \|x_k x_j\| \right)$$

$$C[j-1, j] = \underset{k}{\text{best}} \left(\sum_{\ell=k+1}^{j-2} \|x_\ell \, x_{\ell+1}\| + \|x_k \, x_j\| + \underset{\text{reversed}}{C[k, k+1]} \right)$$