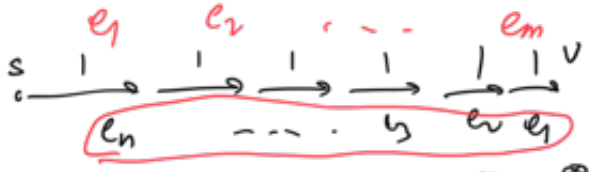
$d[v] \le d[u] + w_e \le D_{i-1}[u] + w_e$

$= D_i[v]$



0 1 2 . . . n.

0	∞	∞	∞	∞	∞	∞	∞	
0	1	∞	. . .				∞	n
0	1	2	∞	∞	.	.	∞	⋮
⋮								n-1
0	1	2	.	.	.		n-1.	

×

Greedy Alg.: Computational problem : min/max some quantity

"Optimal solution": a solution which achieves the min/max

Coin changing problem : | M : find an exact change for M using as few coin as possible

values/denomination : $c_1, c_2, c_3, \ldots, c_k$ (infinite supply)

pick the largest coin $< M$ c_i

and now, $M - c_i$: optimal?

Ex : $c_1 = 1$, $c_2 = 7$, $c_3 = 10$

$M = 14$ 2 coins of 7

10, 1, 1, 1, 1 5 coins.

×

nlyn.

$c_1 < c_2 \le \ldots \le c_n$

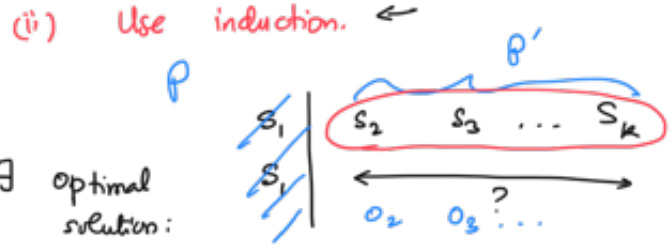
$c_1 | c_2, c_2 | c_3, \ldots, c_i | c_{i+1}$

then greedy is optimal.

How do we prove that a greedy alg. is correct?

✓ (i) The first step taken is correct! There is an optimal solution that agrees with the alg. on the first step. ✓ "Exchange Argument"

✓ (ii) Use induction. ← ✓



After taking the first step, the problem become the same "type" of problem.

optimal solution O where the fist step is also s_1.

P': problem where we have already taken the fist step.

$o_2, o_3, \ldots$: also a solution for P'

$o_1, s_2, \ldots, s_k$ is only better than $o_1, o_2, o_3, \ldots$

"
s_1

Coin-changing problem: $c_1, c_2, \ldots, c_k$ $c_1 | c_2, c_2 | c_3 \ldots$

"
1

10, 7, 1

M : pick the largest coin $c_i < M$. $M - c_i$

$\Rightarrow n = k$

Claim: the greedy alg. is optimal.

Pf: Induction on M.

$\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_n$

✓ $O_1 \geq O_2 \geq \dots \geq O_k$

Exam. ($n \leq k$.)

(i) $\alpha_1 = O_1$ ← first step.

(ii) Both $\alpha_2, \dots, \alpha_n$ and $O_2 \dots O_k$ ($k-1$) are finding change $M - \alpha_1$ ↞ opt. substructure property

By induction, $(n-1) \leq (k-1)$ ✓

$\Rightarrow n \leq k$ = → $n = k$

$M - C_1$

$\alpha_1\ \alpha_2\ \alpha_3 \dots \alpha_n$

(5) 5, 5, 1, 1, 1.

$O_1, O_2, \dots, O_k$ k

5

$n \leq k$?

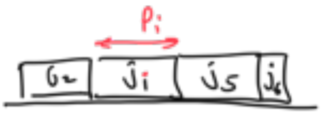
$O_1 \dots O_k$ → M

$O_2 \dots O_k$ → $M - C_1$

— × ————

Weighted Completion Time Scheduling:

jobs	$j_1, j_2, \dots, j_n$.
size	$p_1, p_2, \dots, p_n$
weight	$w_1, w_2, \dots, w_n$



schedule: an order in which to process the jobs.

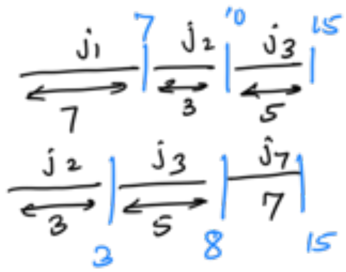
C_j: completion time of the job

Goal: $\min \sum_{i=1}^{n} C_{j_i} \cdot w_{j_i}$ ✓

j_1, j_2, j_3

7, 3, 5

1 1 2

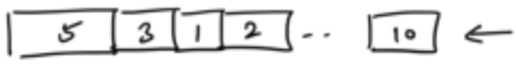


$3 \times 2 + 8 \times 1 + 15 \times 1$

Greedy Alg: $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq \dots \geq \frac{w_n}{p_n}$.

$\frac{w_i}{p_i}$

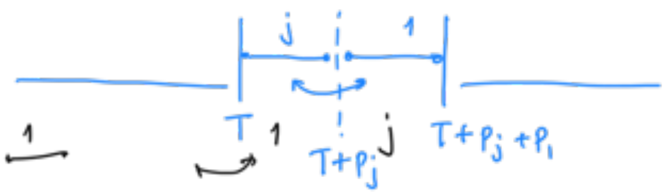
(i) There is an opt alg. which is also processes job 1 first.



1, 2, 3, ..., 10

"Exchange argument": consider an opt. solution. :O

Modify opt to a new optimal solution.



Before the swap: $(T + p_j) w_j + (T + p_j + p_i) w_i$

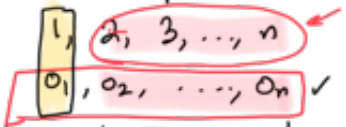
After the swap: $(T + p_i) w_i + (T + p_j + p_i) w_j \leq$

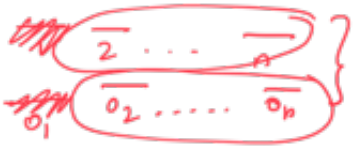
$p_i w_j \leq p_j w_i \qquad \frac{w_i}{p_i} \geq \frac{w_j}{p_j}$

(ii). Suppose the greedy alg. is optimal when there are $< n$ jobs.

Suppose we have n jobs $\quad \frac{w_1}{p_1} \geq \dots \geq \frac{w_n}{p_n}$

By step (i), there is an opt. solution $o_1, \dots, o_n$ where $o_1 = 1$





By induction hypothesis:

weighted comp. $(2, 3, \dots, n) \leq$ weighted comp $(o_2, \dots, o_n)$ ← by I.H.

Weighted comp $(1, \dots, n)$ = weighted $(2, 3, \dots, n)$ + $[w_2 p_1 + \dots + w_n p_1]$

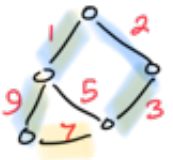
$\wedge\!\vee$

Weighted comp $(o_1, \dots, o_n)$ = weighted $(o_2, \dots, o_n)$ + $[w_2 p_1 + \dots + w_n p_1]$

(I.H.)

Minimum Spanning Trees: $G = (V, E)$, e: wt ← min

connected.



Goal: pick a spanning tree of min. total wt.

Greedy Alg: $w_{e_1} \leq w_{e_2} \leq \dots \leq w_{e_m}$ ←

$T \leftarrow \phi$

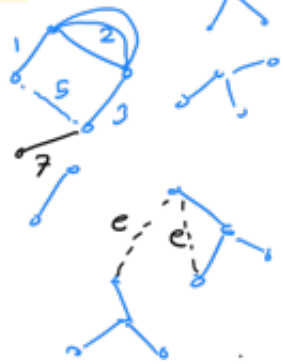
repeat {

if adding e_i to T does not create a cycle

pick e_i and add it to T

}

- Implement? nlogn time
- Disjoint sets.



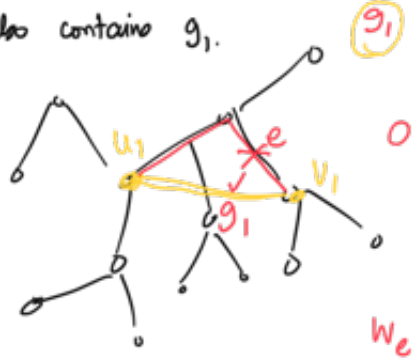
Why is this optimal?

$w_{g_1} \leq w_{g_2} \leq \dots \leq w_{g_{n-1}}$.

(i) Let us say that Greedy: $g_1, g_2, \dots, g_{n-1}$: G

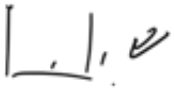
Opt: $o_1, o_2, \dots, o_{n-1}$: O

∃ opt. soln. also contains g_1.



O does not g_1.

$g_1 = (u_1, v_1)$



$w_e \geq w_{g_1}$

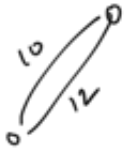
Add g_1 to the soln O: creates a cycle. Remove any other edge from this cycle.

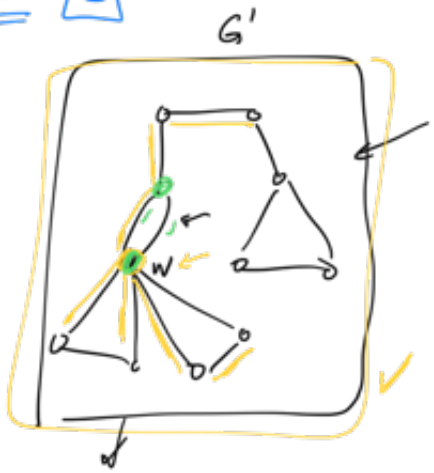
So we get a new opt. soln which picks g_1.

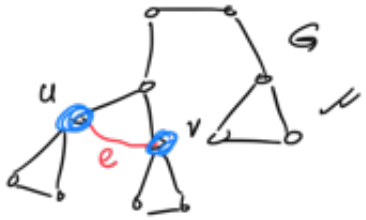
There is an optimal solution which picks

pick e. "0"

(ii) Contract e ← cheapest edge.







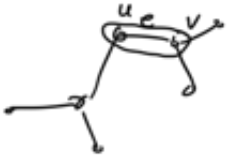
- Contract an edge $e=(u,v)$
  - replace u,v by a new vertex w.
  - replace all edges (u,x) or (v,x) by (w,x) [except for (u,v)].
    all edges between u,v.

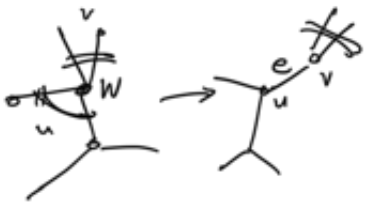
Suppose there is a spanning tree T in G which contains e.

Then, T_e: tree obtained by contracting e.

T_e is a spanning tree in G'.

Conversely, if T' is a spanning tree in G'. Then $T'+e$ is a spanning tree in G. ✓





G.

Running the greedy alg. on G after picking e is same as running greedy alg. on G'.

$e_2, e_3, \ldots, e_k \quad e_{k+1}$ ↑

Greedy: $g_1, \ldots, g_{n-1}$ ‖ $g_1 = o_1 = e$.

Opt: $o_1, \ldots, o_{n-1}$

e

$g_2, \ldots, g_{n-1}$ is the spanning tree produced by greedy on G'

$o_2, \ldots, o_{n-1}$ ✓ also a spanning tree in G' =

By IH: $\text{cost}(g_2, \ldots, g_{n-1}) \le \text{cost}(o_2, \ldots, o_{n-1})$

$+ \text{cost}(g_1) \qquad + \text{cost}(o_1)$

$\Rightarrow$ cost (greedy) on G $\le$ cost (opt) on G ✓

For example:

Huffman encoding:-

A A B A A C A A B C A A A D ← given a text/string

↓ encode as binary string.

Σ : alphabet

e.g. $\Sigma = \{A, B, C\}$, k

00 00 01 00 00 10 — A → 00, B → 01, C → 10, D → 11

fixed length encoding. — $2n$

$2^l \geq k$.

- Variable length encoding

A → 0, B → 01, C → 11, D → 10

00 10 00 11 00 ...

AA

two different strings which have the same encoding

A C A C' — 0 11 0 11

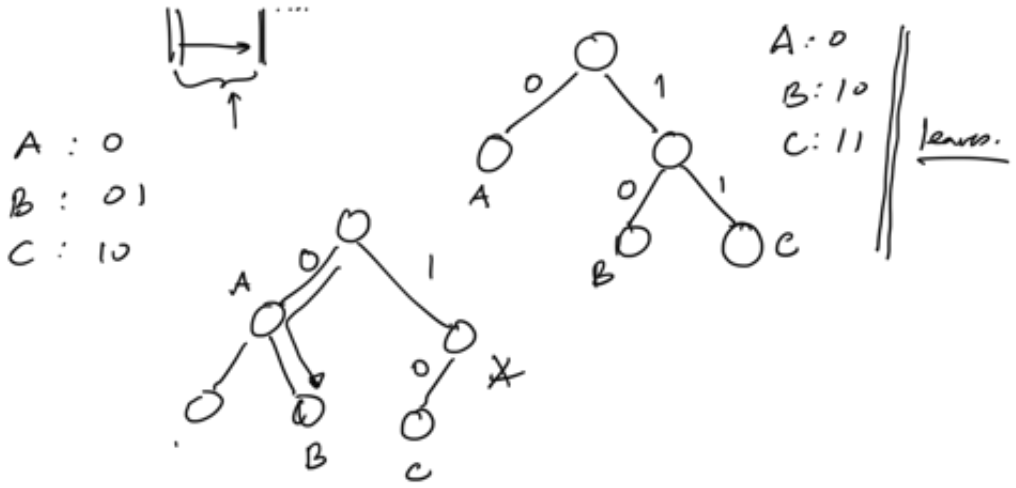
B D C — 01 10 11

prefix-free property: $l_1, \ldots, l_k$

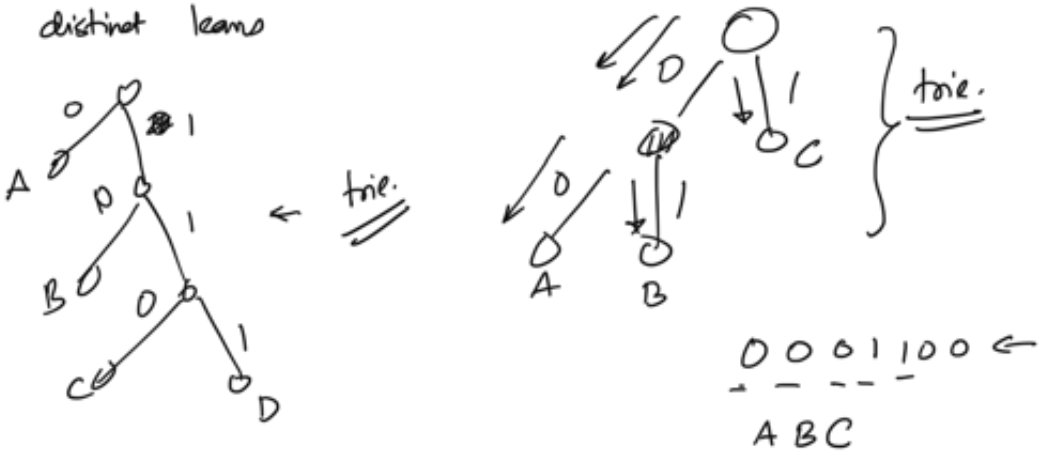
encoding(l_1), encoding(l_2), .., encoding(l_k) : none of these is a prefix of the other.

A : 0, B : 01, C : 10

A : 0, B : 10, C : 11 — leaves.



A labelled binary tree where all the letters appear at distinct leaves



0 0 0 1 1 0 0 ←

A B C

Q : Given a string, find a variable length encoding (trie) with min. length of the encoding

$a_1, \ldots, a_k$

$l_1\ l_2\ l_k$

Length of the encoding of th

$\text{string} = \sum_{i=1}^{n} l_i \cdot f_i = \sum_{i=1}^{n} \text{depth}(a_i) \cdot f_i$

l_i = depth of a_i in the trie

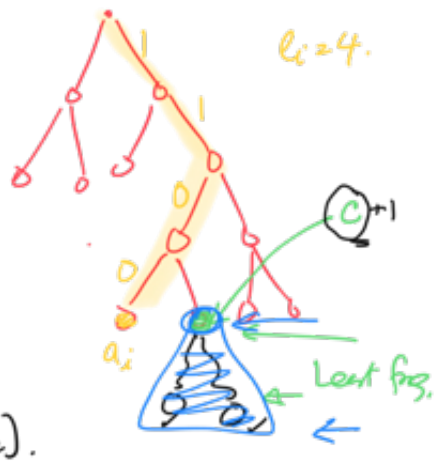
f_i ← how many times a_i appears in the string (frequency).

110	10	111	0
A	B	C	D
10	50	30	70

10		0	11		1	0
↑	B	D	E	← →	F	D
	50	70	40		90	70

C_1: $\text{depth}(C_1) \cdot f_{C_1}$

C_2: $\text{depth}(C_2) \cdot f_{C_2}$



Alg: Let c_1, c_2 be the two letters with min. freq (f_{c_1}, f_{c_2}).

if only 2 letters 0, 1

↓

Replace c_1, c_2 by a new letter $\underline{c}$ $(f_{c_1} + f_{c_2} = f_c)$. 1101 1101

Solve the smaller problem. Let us say the encoding c is σ. $(\sigma = 01110)$ 0, 1

encoding for c_1, c_2 are $\underline{\sigma 0}, \underline{\sigma 1}$

Earlier: $f_{c_1} \cdot l_1 + f_{c_2} \cdot l_2$ (where $l_1 = l+1$, $l_2 = l+1$)

depth of $c = l$

Now: $(f_{c_1} + f_{c_2}) \cdot l$

Diff = $\boxed{f_{c_1} + f_{c_2}}$ ← does not depend on the tree.

01	00	10	11	←	0	10	11	←	0	1
A	B	C	D		E	C	D		E	F
1	1	1	1		2	1	1		2	2

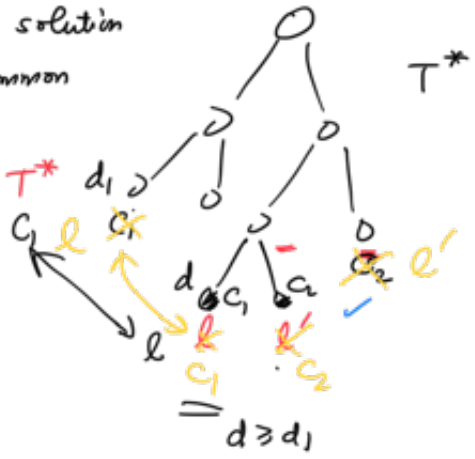
— x —

(i) There is an opt. solution which "matches" with the alg. on the first step.

(ii) Apply induction

(i) Greedy: first combine c_1, c_2 into a new letter c.

We want to prove that there is an opt. solution where the leaves c_1, c_2 have a common parent.



why? Let l be the deepest leaf in T^*

$\Rightarrow \sum_i \text{depth}(a_i) \cdot f_{a_i}$

$$\begin{cases} l \leftarrow d \\ c_1 \leftarrow d_1 \end{cases} \qquad \begin{cases} l \leftarrow d_1 \\ c_1 \leftarrow d \end{cases}$$

$$f_l \cdot d + f_{c_1} \cdot d_1 \geq f_l \cdot d_1 + f_{c_1} \cdot d$$

because

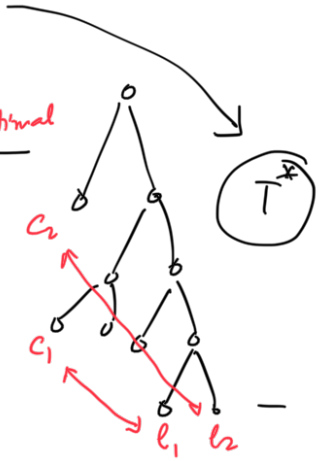
$$f_l \cdot d + f_{c_1} \cdot d - f_l \cdot d_1 - f_{c_1} \cdot d \geq 0$$

$$\Leftrightarrow (\underbrace{f_l - f_{c_1}}_{\geq 0})(\underbrace{d - d_1}_{\geq 0}) \geq 0$$

— x —

16/9/21

(i) We showed that there is optimal solution T^* where c_1, c_2 have a common parent.



(ii) Induction proof: Assume that the greedy alg. is optimal when there $\leq k-1$ letters.

k letters: $\{a_1, \dots, a_k\} = \Sigma$

At the first step: the greedy alg. combines letters $c_1, c_2 \to c$.

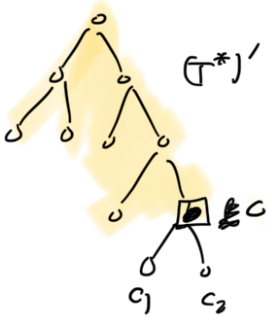
By (i), there is an opt. solution T^* where c_1, c_2 have a common parent

$\Sigma' = (\Sigma - \{c_1, c_2\}) \cup \{c\}$: $k-1$ letters.

By induction hypothesis, the greedy alg is opt. for Σ'.

$(T^*)'$: T^* with c_1, c_2 removed and the common parent labelled c.

Then, $(T^*)'$ is also a solution for Σ'!



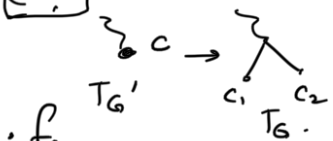
By IH: the encoding length of greedy on Σ' $\leq$ encoding length of $(T^*)'$

$+ f_{c_1} + f_{c_2}$ $+ f_{c_1} + f_{c_2}$

encoding length of Σ in T_G $\leq$ encoding length of T^*

T_G: greedy solution tree

$(T_G)'$: greedy tree for Σ'.



$T_G' \to c$

$\sum_{\ell \in \Sigma'} \text{depth}(\ell) \cdot f_\ell$ (ℓ in T_G')

$\text{depth}(c) \cdot f_c$

$= \text{depth}(c) \cdot f_{c_1} + \text{depth}(c) f_{c_2}$

$\sum_{\ell \in \Sigma} \text{depth}(\ell) \cdot f_\ell$ (ℓ in T_G)

$\text{depth}(c_1) \cdot f_{c_1} + \text{depth}(c_2) \cdot f_{c_2}$

diff $= f_{c_1} + f_{c_2}$ ∵

$\text{depth}(c_1) = \text{depth}(c_2) = \text{depth}(c) + 1$

Next topic: AMORTIZED COMPLEXITY.