BFS (s)

repeat till Q is empty.

v = Deque(Q)

for all neighbours w of v (out-neighbours) {

(visited[w] = false)

Enqu(w), visited[w] = true. ✓

Level(w) = Level(v) + 1 ;

p[w] = v;

parent inf.

}

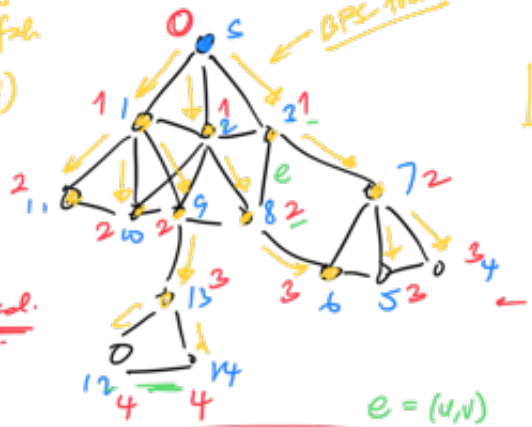
for v = 1, ..., n

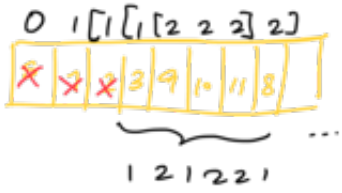
if visited(v) = false

start BFS(v)

$\sum_v (deg(v)+1) = |V| + |E|$.

Level(v) = distance from s in the BFS traversal.

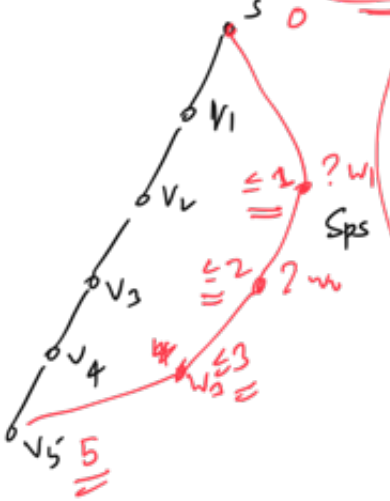




How do we prove that BFS tree is a shortest path??

e = (u,v)

Any (u,v) ∈ E : $|level(u) - level(v)| \le 1$

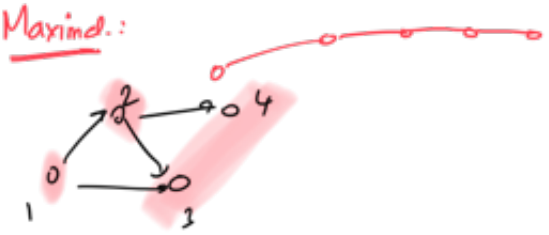


5 edges s to v_5 in the BFS tree.

Sps a there is a shorter path from s to v_5 : 4 edges?

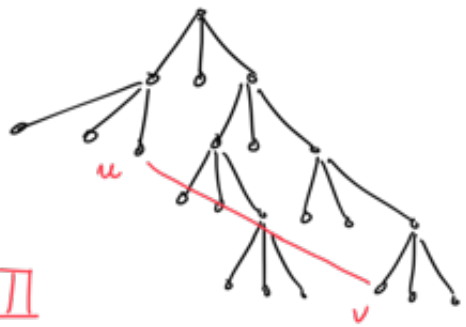
⇒ BFS tree is a shortest path tree.

Maximal: Longest ≤ k.



Property of BFS tree. :

(*) If (u,v) ∈ E, level(u), level(v) differ by at most 1. ✓



Q

(*) ⇒ the Q always has the following property:

(P) ✓ – the levels of the vertices are in ascending order from front to rear.
✓ – levels of any two vertices on Q differ by at most 1.

x x x x x -- x || 1 2 3 4 x

x x x x x -- x+1 -- x+1

Prove by induction (on the # steps of the BFS alg.):

(i) (P) Satisfied at the beginning ✓

IH (ii) Suppose (P) is satisfied at some step t, then (P) is also satisfied at the next step.

(i) s | ✓ (0)

(ii) v | x | x | .. | x | x+1 ✓ ✓

or | x | x | .. | x | x+1 | .. | x+1 ✓

Deque(v), Enque some of the neighbours of v.

⊛: Suppose $(u,v) \in E$. Let us say

→ u is Dequeued before v. at some time t.

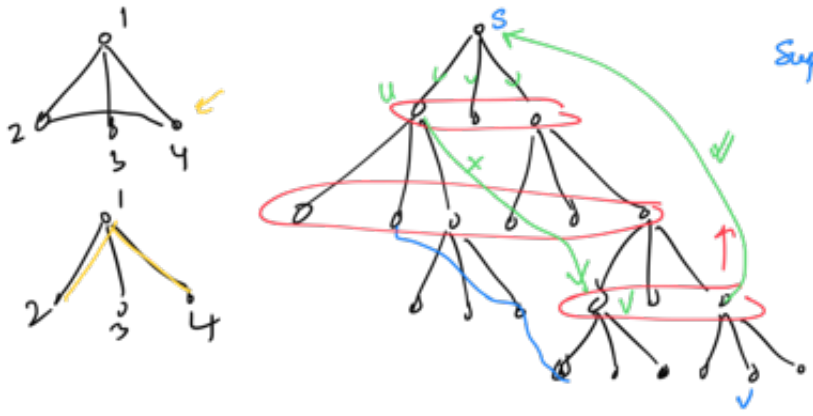
Q: | u | ... | v |

x x ... x x+1 ... x+1

Where is v? (i) v is in the Q at time t → (P) ⇒ level(v) = level(u) or level(u)+1

or (ii) v has not been visited yet → alg. will add v to the Q at level(v) = level(u)+1.

∴ level(v) = level(u) or level(u)+1



Suppose v is a vertex at level ℓ of the BFS tree.

s — v1 — v2 — ... — v

Let P be any path in the graph from s to v.

s — w1 — w2 — w3 — ... — v.

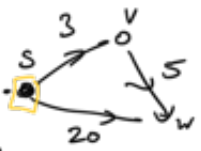
v0 ≤1 ≤2 ≤3 wk

level(wi) ≤ i level(v) ≤ k. ⇒ k ≥ ℓ.

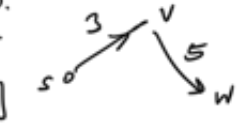
Directed Graph: (u,v) 10 u → v 1

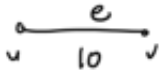
level(v) ≤ level(u) + 1 ✓

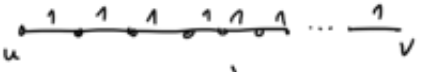
Shortest Path in Weighted (Directed) Graphs: Dijkstra's.



e — Each edge e has a weight $w_e \geq 0$.







$w_e \in [0, L]$ / 10^8

∀e: $w_e = 1$ → BFS. ✓

sub-divide e into w_e edges and then run BFS.

— | V | "

BFS visits vertices in order of levels. distance from s. ✓

$s, v_1, v_2, v_3, v_4, \ldots, v_n$

Dijkstra's alg. will also "visit" vertices in order of increasing distance from s:

$s, v_1, v_2, v_3, \ldots, v_n$ ← vertices arranged in inc. order of distance from s.

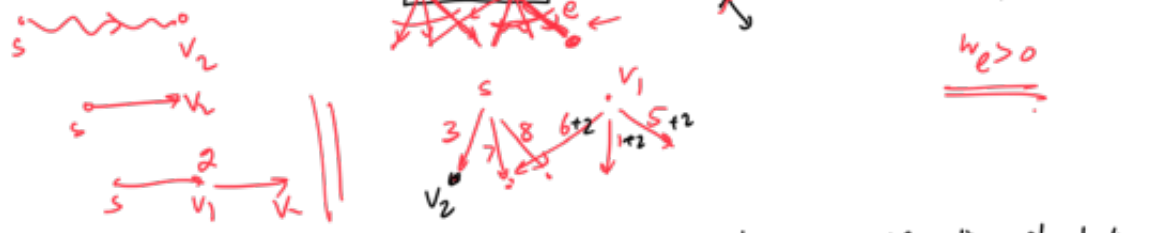
$s, v_1, v_2, \ldots, v_i$
$0 \quad d_1 \quad d_2 \quad \cdots \quad d_i$

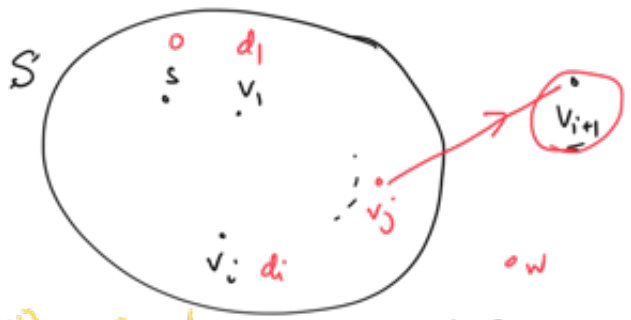
v_{i+1} ??

v_2 ?

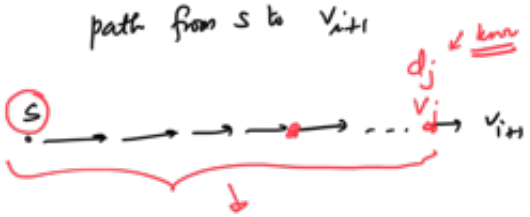
$w_e \geq 0$

$w_e > 0$





Let us consider the shortest path from s to v_{i+1}



$\in \{v_1, \ldots, v_i\}$

then the length of this path $= d_j + w(v_j, v_{i+1})$.

$\delta^+(S) = \{(u,v) : u \in S, v \notin S\}$

visited[v]

Let $S \leftarrow \{s\}$. $D[s] = 0$

While (S is not all of V) {

store the value

Consider all edges in $\delta^+(S)$: $e = (u,v)$ $u \in S, v \notin S$

$\rightarrow X_e = D[u] + w(e)$ ←

pick the min. value x_e : e^*

Suppose $e^* = (u,v)$ visited[v] = true

then add v to S, $D[v] = x_{e^*}$

}

AVL tree.

deg(v) log m

n

$O(nm)$.

$O(m)$ ✓

After i iterations of the while loop

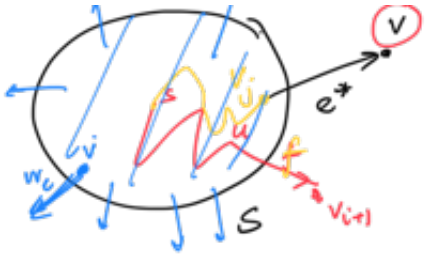
$S = \{s, v_1, \ldots, v_i\}$ and $D[v]$ will be the shortest path distance from s to v for all $v \in S$

Base Case: $S = \{s\}$. $D[s] = 0$ ✓

Induction Case: $S = \{s, v_1, \ldots, v_i\}$, $D[v_1], \ldots, D[v_i]$ ✓

$\sum_v d_v(v) \log n$

$= |E| \log |V|$.



Claim: $v = v_{i+1}$ and $x_{e^*} = D[v]$

Let P be the shortest path from s to v_{i+1}.

$x_f = D[u] + l(f) = D[v_{i+1}]$

$x_{e^*} = D[v_j] + l(e^*)$ ← length of a path from s to v

$x_{e^*} \le x_f \Rightarrow v = v_{i+1}$

$x_{e^*} = D[v_{i+1}]$ //

Let $S \leftarrow \{s\}$

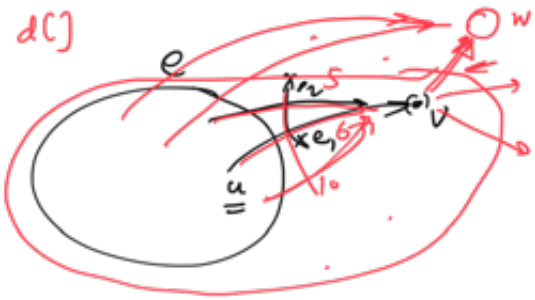
while ( ) { $d[v] = \infty$

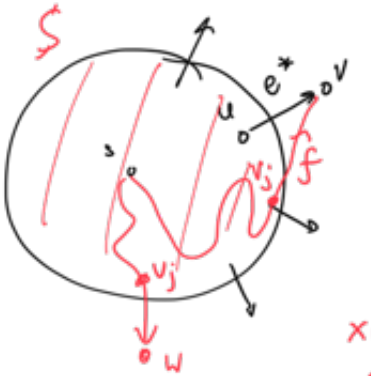
For every edge $e = (u,v)$ in $\delta^+(S)$ $d[v] \leftarrow \min(d[v], D[u] + w_e)$

pick the vertex $v \notin S$ with min $d[v]$ value.

Add v to S, $D[v] \leftarrow d[v]$

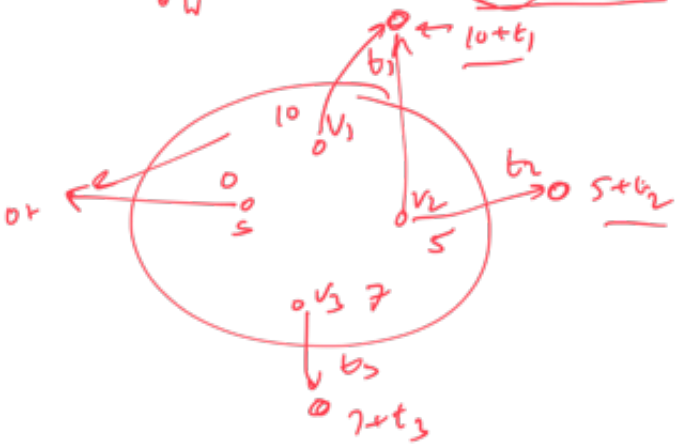
}





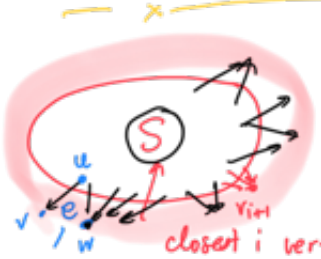
$D[u] + w(e^*) = x(\cdot)$ $x(e^*) \le x(f)$ ✓

$x(f) = D[v_j] + w(f)$ ✓



6/9/21.

Dijkstrak alg.



$d(v)$.

AVL tree

x_e values.

$\delta^+(S)$

$s, v_1, \ldots, v_i$ v $D[\,]$.

x_e

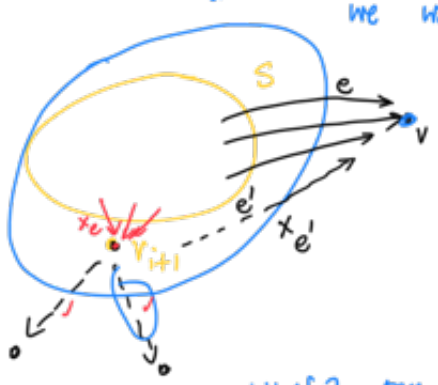
closest i vertices to S

$m = |E|$, $n = |V|$.

$\min$ $(w_e + D[u])$ ← $\min$ $\min$ $[w_e + D[u]]$

// $e \in \delta^+(S)$ $v \notin S$ $e:(u,v)$

$d(v) = \min_{e:(u,v),\ u\in S} [w_e + D[u]]$ $\quad d(v)$



we want to know $\min_{v\notin S} d(v)$

[AVL tree store x_e for all $e \in \delta^+(S)$. $\deg(v_{i+1})$. $\lg m$ $m \lg m$ $\le$ $m \lg n$.]

$d(v)$? $\quad \min_{e:(u,v),\ u\in S} [w_e + D[u]]$

update ?

$d(v) = \min [d(v),\ w_{e'} + D[v_{i+1}]]$

visited[s] = true,

$D[s] = 0$ initially $d[v] = \infty$, $v \neq s$

repeat {

Let u be the vertex visited[u] = false and d(u) is min. $\quad$ m lgn.

visited[u] = true; $D[u] = d(u)$ ← stored in AVL/Heap.

for all edge $e:(u \to v)$ where visited[v] = false

$d[v] = \min(d(v),\ w_e + D[u])$; $p[v] = u$

}

mlgn / lgn / Heap / AVL / n

Fibonacci Heap: $m + n\lg n$ $\quad$ Heap: $m \lg n$

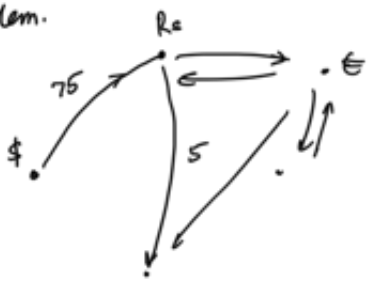
$(m \ge n-1)$.

What happens when we could be -ve. edge lengths !

(i) $w_e = -1$ for all edges : longest path $\quad$ min. $\quad -|P|$

NP-Hard problem.

(ii)



currencies.

$\alpha_1 + \alpha_2 + \alpha_3$

$w = -\log \alpha_i$

$\alpha_1 \cdot \alpha_2 \cdot \alpha_3 > 1$

is there a cycle whose length is negative ?

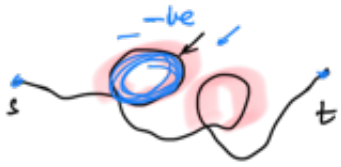
Q : want to detect if there is a cycle such that $\prod_{i\in C} \alpha_i > 1$?

$\sum_{i\in C} \log(\alpha_i) > 0$ or $\sum_{i\in C} -\log(\alpha_i) < 0$ $\quad$ (+ −)

s, t

shortest $s-t$ path

shortest $s-t$ walk $-\infty$



$-\infty$

if -ve edge present, then length of shortest $s-t$ walk could be $<$ length of shortest $s-t$ path

-ve cycles

As long as, there is no negative cycle, shortest walk $\equiv$ shortest path

[There are polynomial time alg. to find the shortest $s-t$ path]

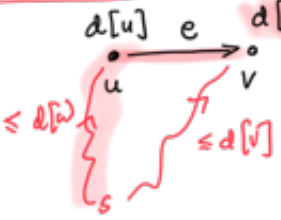
Bellman-Ford Alg.: $O(mn)$. ✓ $O(m \log n)$ ✓

Maintain $d[v]$: estimate of shortest path from s to v.

$D[v]$: length of the shortest "

$d[v] \geq D[v]$ always.

Initially $d[s] = 0$, $d[v] = \infty$.



$D[v] \leq d[u] + w_e$

if $(d[v] > d[u] + w_e)$ then $d[v] = d[u] + w_e$ ⊛.

repeat :

if there is an edge $e = (u,v)$ s.t.

$d[v] > d[u] + w_e$ ✓ ←

update

$d[v] = d[u] + w_e$ ←

until no such edge can be found. ✓

$D[v] \leq U$ → $O(nU)$

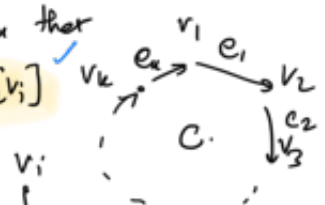
$O(mn)$. ✓

This procedure will run in an infinite loop if there is a negative cycle! When the procedure stops: $d[v] \leq d[u] + w_e$ $\forall e = (u,v)$

Pf: Suppose not. Suppose it terminates.

$\forall e_i$: it must be the case that

k inequalities → $d[v_{i+1}] \leq w_{e_i} + d[v_i]$ ✓



add all these inequalities:

$d[v_2] \leq w_{e_1} + d[v_1]$ ✓

Claim: For all w, $d[w] \geq D[w]$

Pf: By induction on # steps.

Base case: true initially?

⊛ happens: is this still true?

we reduce $d[v]$

By IH, P has length $\leq d[u]$ ($\because d[u] \geq D(u)$).

$D[v] \leq d[u] + w_e$

$w_{e_1} + w_{e_2} + \cdots + w_{e_k} < 0$ ⊛ ✓

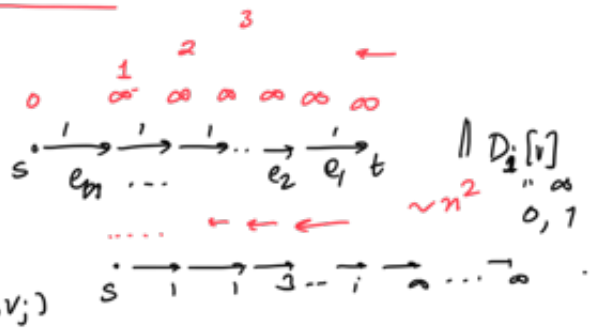
Sps $d[v_{i+1}] > d[v_i] + w_{e_i}$

$d[v_3] \le w_{e_2} + d[v_2]$

$d[v_1] \le w_{e_3} + d[v_3]$

$0 \le w_{e_1} + w_{e_2} + w_{e_3} + \cdots + w_{e_m}$ ✓

$e_1, e_2, \ldots, e_m$



outer for $i = 1 \ldots n-1$ — $n-1$

for $j = 1 \ldots m$ — m

mn

Suppose $e_j = (u_j, v_j)$

if $d[v_j] > d[u_j] + w_{e_j}$

$d[v_j] = d[u_j] + w_{e_j}$ } (1) $O(nm)$.

if there is an edge $e=(u,v)$

$d[v] > d[u] + w_e$ then during negative cycle.

$D_i[v]$: shortest path from s to v using $\le i$ edges.

$$D_0[v] = \begin{cases} \infty & \text{if } v \neq s \\ 0 \end{cases}$$

Claim: After i iterations of the outer for-loop

$d[v] \le D_i[v]$.

$i = n-1$

$d[v] \le D_{n-1}[v]$

$\vdots$

$D[v]$

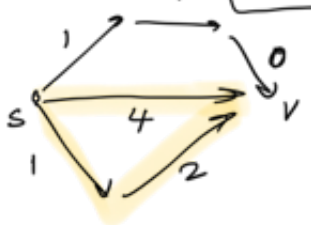
$d[v] \ge D[v]$

Pf: induction on i.

$i = 0$? $d[v] = \infty$, $d[s] = 0$

$D_0[v] = \infty$, $v \neq s$

$D_0[s] = 0$



$D_0[v] = \infty$

$D_1[v] = 4$

$D_2[v] = 3$

$D_3[v] = 2$

$\vdots$

$D_{n-1}[v] = D(v)$

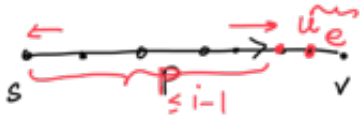
$s \longrightarrow v$

Sps this statement is true for $i-1$.

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

v i-th itn.

$d[v] \le D_i[v]$ ⊛



Let P be the shortest s–v path of length at most i

length of P = (part of P from s–u) + edge e

$w(P) = D_i[v]$

$D_i[v] = D_{i-1}[u] + w_e$

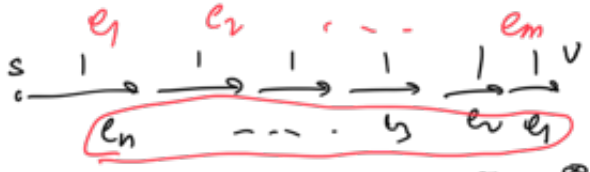
At the beginning of this itn,

$d[u] \le D_{i-1}[u]$ ✓

when we look at this e during this itn. of the for loop:

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

$= D_i[v]$



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.

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

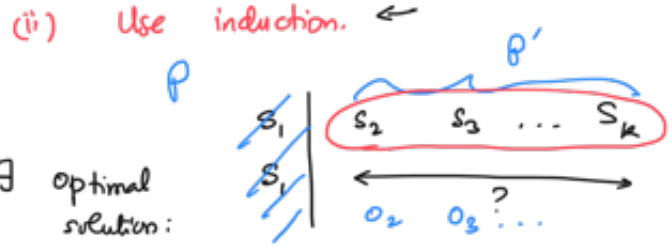
$c_1 | c_2, c_2 | c_3, \ldots, c_i | c_{i+1}$

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.

$\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_n$ ✓ $o_1 \geq o_2 \geq \dots \geq o_k$

(i) $\alpha_1 = o_1$ ← first step.

(ii) Both $\alpha_2, \dots, \alpha_n$ and $o_2 \dots o_k$ are finding change $M - \alpha_1$

By induction, $(n-1) \leq (k-1)$ ✓

$\Rightarrow n \leq k$ $\Rightarrow n = k$

Exam.

$n \leq k$.

M-G.

$k-1$

$o_2 \dots o_k$ ← opt. substructure property

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

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

$o_1, o_2, \dots, o_k$ k

5

M-G

$n \leq k$?

C_1

$O_1 \dots O_k \to M$

$O_2 \dots O_n \to M-G$

— × —

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. $\sum_{i=1}^{n} C_{j_i} \cdot w_{j_i}$ ← ✓

p_i

j_2 j_i j_5 j_k

j_1, j_2, j_3

7, 3, 5

1 1 2

j_1 (7) | j_2 (3) | j_3 (5) | 15

7 10

j_2 (3) | j_3 (5) | j_7 (7) |

3 8 15

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

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 process job 1 first.

| 5 | 3 | 1 | 2 | .. | 10 | ← ✓

1, 2, 3, ..., 10

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

1 1 1 t

j i k 1

S

Modify opt to a new optimal solution.

j i 1

1 T 1 $T+p_j$ j $T+p_j+p_1$

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

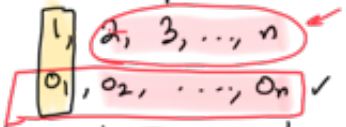
After the swap: $(T + p_i) w_i + (T + p_j + p_i) w_j \leq$

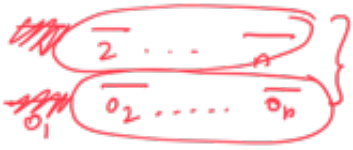
$p_i w_j \leq p_j w_i \qquad \frac{w_i}{p_i} \geq \frac{w_j}{p_j}$

(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 $(2, 3, \dots, n)$ + $[w_2 p_1 + \dots + w_n p_1]$

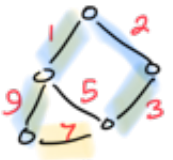
$\wedge\!\vee$

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

$T \leftarrow \phi$

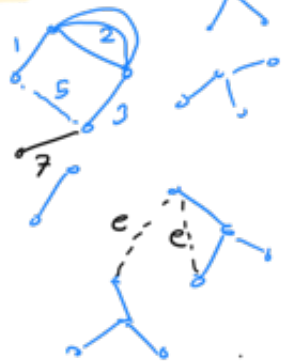
repeat {

if adding e_i to T does not create a cycle

pick e_i and add it to T

}

- Implement? nlogn time
- Disjoint sets.



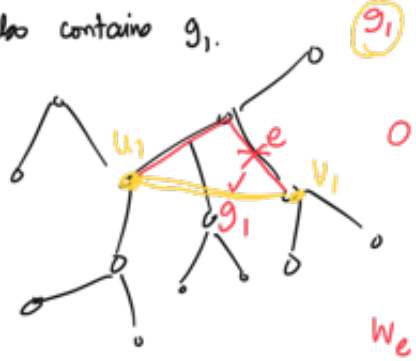
Why is this optimal?

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

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

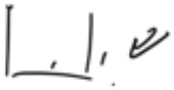
Opt: $o_1, o_2, \dots, o_{n-1}$: O

∃ opt. soln. also contains g_1.



O does not g_1.

$g_1 = (u_1, v_1)$



$w_e \geq w_{g_1}$

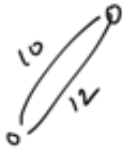
Add g_1 to the soln O: creates a cycle. Remove any other edge from this cycle.

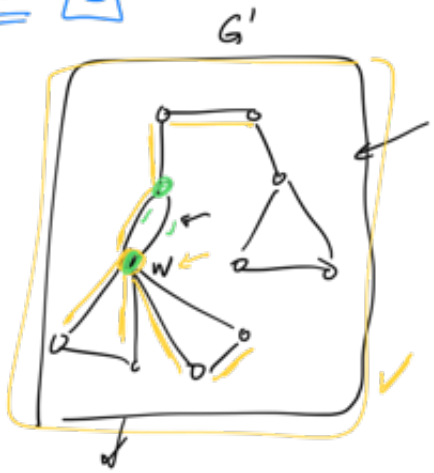
So we get a new opt. soln which picks g_1.

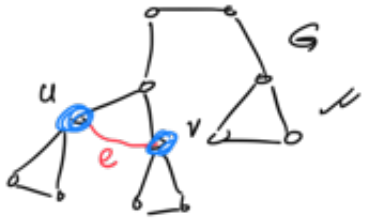
There is an optimal solution which picks

pick e. "0"

(ii) Contract e ← cheapest edge.







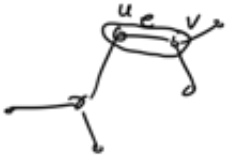
- Contract an edge $e=(u,v)$
  - replace u,v by a new vertex w.
  - replace all edges (u,x) or (v,x) by (w,x) [except for (u,v)].
    all edges between u,v.

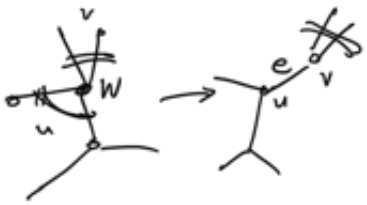
Suppose there is a spanning tree T in G which contains e.

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

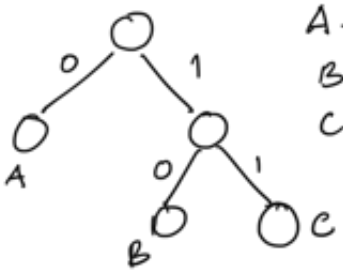
A C A C — 0 11 0 11

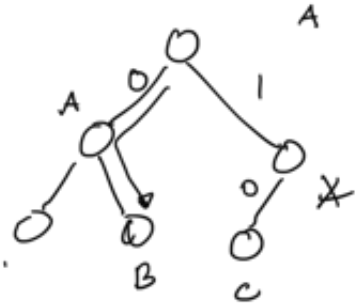
B D C — 01 10 11

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.

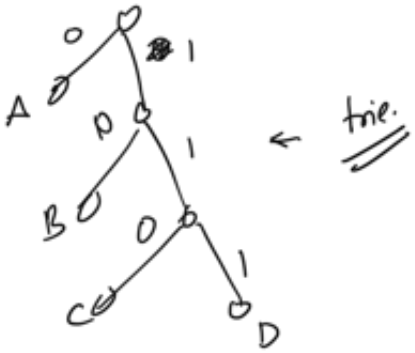
A : 0, B : 01, C : 10

A : 0, B : 10, C : 11 — leaves.





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



← trie.

trie.

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

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

ℓ_i = how many times a_i appears in stirg (frequency).

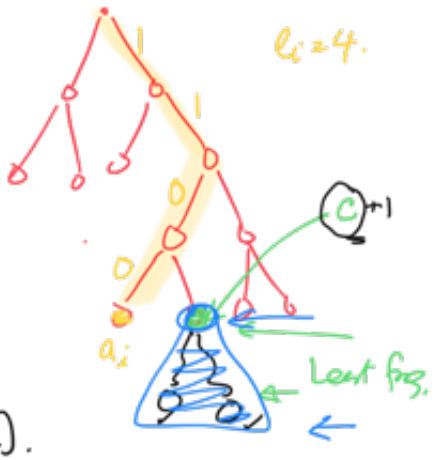
depth of a_i in the trie

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

B 50 (10) D 70 E 40 (11) F 90 D 70

C_1: depth(C_1) · f_{C_1}

C_2: depth(C_2) · f_{C_2}



$\ell_i = 4$.

Alg: (if only 2 letters 0,1)

Let C_1, C_2 be the two letters with min. freq (f_{C_1}, f_{C_2}).

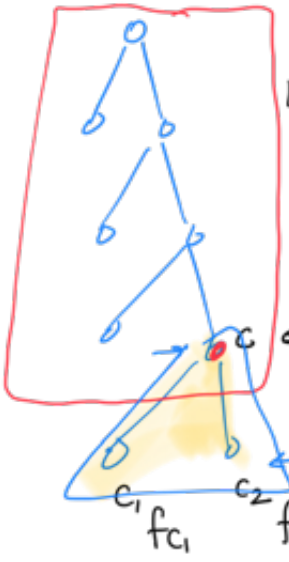
↓

Replace C_1, C_2 by a new letter C (1101 1101)

$(f_{C_1} + f_{C_2} \equiv f_C)$.

Solve the smaller problem. Let us say the encoding of C is σ. ($\sigma = 01110$)

encoding for C_1, C_2 are $\sigma 0, \sigma 1$ (0, 1)



Earlier: $f_{C_1} \cdot \ell_1'' + f_{C_2} \cdot \ell_2''$ ($\ell+1$)

Now: $(f_{C_1} + f_{C_2}) \cdot \ell$

Diff = $f_{C_1} + f_{C_2}$ ← does not depend on the tree.

01	00	10	11
A	B	C	D
1	1	1	1

←

0	10	11
E	C	D
2	1	1

←

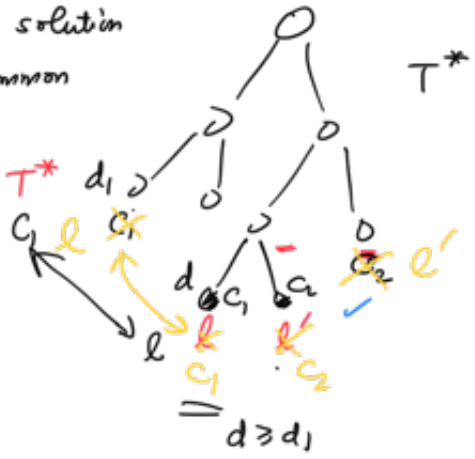
0	1
E	F
2	2

(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 —