MESSAGES | PHONE CALLS

WEDNESDAY NOVEMBER 10

Asymptotic Analysis : not a well defined

Program → What is running time ?
- i/p not specified
- s/m not specified
- CPU h/w

+1 n
+2n sum = 0
+2n for i = 1 .. n:
sum = sum + A[i]

+	1ns
compare	0.5 ns
*	...
=	
<>	
A[]	

algo as a sequence of "basic" inst" ⇓ 1 unit of time

- Get an "approximate time": is sec in min, n hours, n dep.. ?
- compare algorithm ?
- how many UNITS of time ? counting : 4n+100, 6n+1 factor 2-3 loss.

only concern with large n.

max. n, A
+1 max = -1
+2n for i = 1 ... n do
+2n if max < A[i]
?? → max = A[i]. (4)
o/p max.

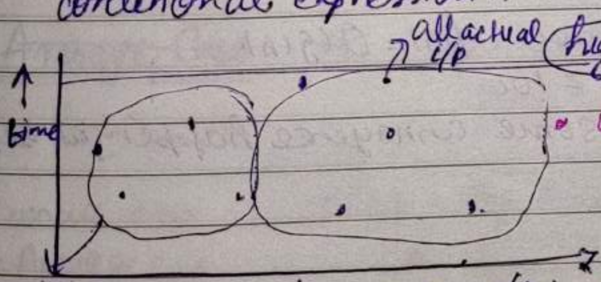
54321, 12345, 1,5.

depend on relative ordering value.

Avg → 2l = $2 \log_2 n$.

$4n+2+2l$: running time may not be expression on n alone.

conditional expression : running time could depend critically on mere



all actual i/p highest

- Worst case analysis (report max value of running time over all i/p of length)
- Best-Case
- average case σ: permutation E(Running time on σ) i/p may come from some other dist-
- difficult to calculate
- some i/p may be more likely

never comes different i/ps of length(n)

1999

JULY	AUGUST	SEPTEMBER	OCTOBER	NOVEMBER	DECEMBER
M T W T F S S WK	M T W T F S S WK	M T W T F S S WK	M T W T F S S WK	M T W T F S S WK	M T W T F S S WK

11 THURSDAY NOVEMBER

MESSAGES	PHONE CALLS

1. # basic instⁿ $\frac{1}{2}n^2 + 6n$
2. worst possible i/p for each n (n^2+1)
3. Compare alg. or understand how running time scales as we raise n

$100n + 20$ (A) $\qquad \frac{n^2}{100} + 6n$ (B)

A is better. (if we only care abt how running time scale as we raise n)

$n \equiv 2n$ $\quad A \equiv 2A$ but $B \equiv 4A$

$T(n)$: expression for running time.

only care abt the larger term & ignore constant

3rd type of approximat^n } A: $4n$ B: $100n$ ⇒ for us both will same.

$100n^2 \log n + \frac{n}{1000} + 1000000n^2$.

$O(n^2 \log n)$.

$O()$ notation.

Class 2: 12/Aug.

Def^n. $O(n)$ notation. Running time, some known fixed

$f(n)$, $g(n)$ are two fⁿ of n, where n is non-ve int^g

we say that $f(n) = O(g(n))$ if

$\exists\, c > 0$ $\quad f(n) \le c\, g(n)$ for all n

eg. $f(n) = 2n$ $\quad g(n) = n$ $\quad 2n \le 2(n)$

$f \quad c \quad g$.

$f(n) = 100n^2 + 50n$ $\quad g(n) = n^2$ $\quad f(n) \le \underline{150}\, g(n)$

$f(n) =$ Why we need to have n_0 in this defn $\quad (n \ge n_0)$

$f(n) = n \quad n \ge 1$

$g(n) = \begin{cases} n & n > 100 \\ 0 & \text{otherwise} \end{cases}$

we still want to have $f(n) = O(g(n))$.

$n_0 = 100$.

These could be some annoyance happening at beg of

$f(n) = O(g(n))$

if $\exists\, c > 0$ $\exists\, n_0 > 0$ such that

$\forall\, n \ge n_0$.

$f(n) \le c\, g(n)$

Defⁿ.

1999

JANUARY							1
M	T	W	T	F	S	S	WK
				1	2	3	1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY							2
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH							3
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL							4
M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY							5
M	T	W	T	F	S	S	WK
31					1	2	18/23
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE							6
M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES | PHONE CALLS

FRIDAY NOVEMBER 12

- $f(n) = n \quad g(n) = n^2$
  $f(n) \neq O(g(n))$ ← by contradict^n
  → suppose there is a c, n_0 s.t. $f(n) \leq c\,g(n) \quad \forall\ n \geq n_0$.
  $n^2 \leq cn \quad \forall\ n \geq n_0$.
  $n \leq c \quad \forall\ n \geq n_0$ ✗

- $g(n) = n$ if n is even, $g(n) = 1$ if n is odd. $f(n) = n$.
  if $f(n) = O(g(n))$
  suppose $\exists\ n_0, c$ s.t. $f(n) \leq c\,g(n) \quad \forall\ n \geq n_0$.
  pick n which is odd $n \geq n_0$ $n \leq c$ ✗

- $f(n) = \Omega(g(n))$ if
  $\exists\ c, \exists\ n$ s.t. $\forall\ n \geq n_0$. $f(n) \geq c\,g(n)$
  $n^3 = \Omega(n)$, $\frac{n}{100} - 50$ is $\Omega(n)$ $n^2 - 10000n - 70$ is $\Omega(n)$

eg. $T(n)$: worst case running time
$T(n) = O(n^2)$ $T(n) = \max\{T(I_1), T(I_2) \ldots\}$
i/p of length n → I_1, I_2 $T(n) = O(n^2)$ $T(n) = \max(T(I_1), \ldots) \geq cn^2$ for some c ↔ there is an i/p I s.t. $T(I) \geq cn^2$

eg. max finding
$T(n)$ = # times instr^n ④ executed.
$T(n) = O(n)$ Base case: 1
$T(n) = \Omega(n)$ consider the i/p 1, 2, 3, .. n

Average Case: Θ $f(n) = \Theta(g(n))$ if
$f(n) = O(g(n))$ & $f(n) = \Omega(g(n))$

- worst case $T(n)$ = Avg time of running ④ is executed.
- Average case A [ ] All permutations.

$\sum$ [# ④ is executed one] → $n!$
(counted by me)

1999

JULY

M	T	W	T	F	S	S	WK
		1	2	3	4		27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST

M	T	W	T	F	S	S	WK
30	31					1	31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER

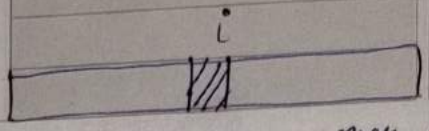
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

13 SATURDAY NOVEMBER

MESSAGES | PHONE CALLS



how many of the n! i/p will be update max when we compare A[i], max

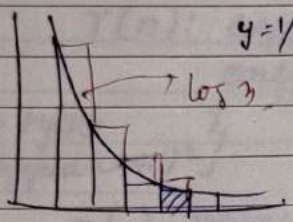
$$\frac{\binom{n}{i} \cdot (n-i)!\,(i-1)!}{n!} = \frac{1}{i}$$

Choose max # of i number

$\sum$ [# times ⊛ is executed] = $\sum_{i=1}^{n}$ [# permutations for which ④ is executed in the i^{th} iteration] / n!

$= \sum_{i=1}^{n} \left(\frac{1}{i}\right)$

$= 1 + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n} = \log_e n + 1$

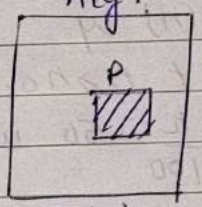


Amortized case:

Alg called P k time.

k T(n).

may be the worst case for sequence of ~~operat~~ calls.



T(n) = worst-case

eg n bit # add 1 to it.

1 0 1 1 0 0 1 1

+ 1

1 0 1 1 0 1 0 0

T(n) = # bit operⁿ

T(n) = O(n). } T(n) = Θ(n)

0 1 1 1 1 1 1 1 T(n) = Ω(n) }

Avg case: all n-bit #s. Running time

x — x — x — 2^n

Suppose we start fⁿ encrement with x=0 & keep adding 1 fr

14 SUNDAY n steps what is the total running time?

n [n] ≤ n^2

n times c/d [n] worst case time.

How often will T(n) be just

LSB = 0

1 n/2

2 n/4

3 n/8

i n/2^i

log n 1999

X should be log n

MESSAGES | PHONE CALLS

MONDAY NOVEMBER 15

$\frac{n}{2} \cdot 1 + \frac{n}{4} \cdot 2 + \cdots \frac{n}{2^i} \cdot i \leq \sum_{i=1}^{\log n} \frac{n}{2^i} \cdot i \leq 3n \sum_{i=1}^{\infty} \frac{i}{2^i} \leq 3$

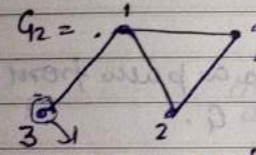
Amortized time complexity: n operations. Total running time of n operations / n
how it is different from Avg case?

Class 3: 16/Aug

graphs: Study 'pairwise' relations.

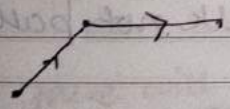
example:
- FB: vertices = people, edges = friends.
- www: vertices = webpage, edge = hyperlink
- Road n/w: vertices = city, edge = highway/roads
- Flight n/w: vertices = airports, edge = flights

$G(V, E)$ - V is a finite set
E = collection of pairs of V
$= V \times V = \{(a,b) \mid a \in V, b \in$
$E \subseteq V \times V$ (directed $(a,b) \neq (b,a)$
undirected $(a,b) = (b,a)$



u
1) Neighbour $\Gamma(u)$ = Set of vertices joined to u by edge
$= \{v : (u,v) \in E\}$ undirected
2) Degree of vertex: $|\Gamma(u)|$
on undirected $\sum_{u \in V} \deg(u) = 2|E|$ (Euler's)

every edge contributes two vertex degree



D.
1) out neighbours $\Gamma^+(u)$ = set of vertex leaving u
$\{v : (u,v) \in E\}$
2) in neighbour: $\Gamma^-(u) = \{v : (v,u) \in E\}$
3) out-degree in degree. $\sum_{u \in V}$ outdegree $(u) = |E|$

(2) Walk / path:
a walk w is a sequence of vertices $v_1 \, v_2 \ldots v_r$ s.t.
$\forall \; i = 1, 2 \ldots r \quad (v_i, v_{i+1}) \in E$

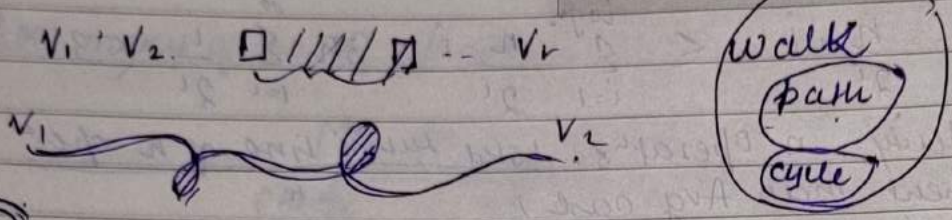
walk: vertices could repeated
path: vertices could not be repeated.
Cycle: walk where only 1st & last are same & others distinct

1999 should have 2 vertex

atleast 2 vertices

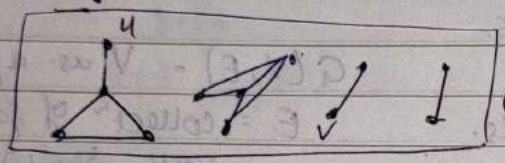
16 TUESDAY NOVEMBER

MESSAGES | PHONE CALLS



(3) Connectivity: undirected:

G is connected: if there is a path from every vertex u to every other vertex.



maximal connected subgraph of G. (something you could not expand)

SubGraph: if $G(V,E)$ is a graph, then $G'(V';E')$ is subgraph of G if $V' \subseteq V$, $E' \subseteq E$.

- Formal defn of connected component:

we define relation on V: $u \sim v$ if there is a path from u to v in G.

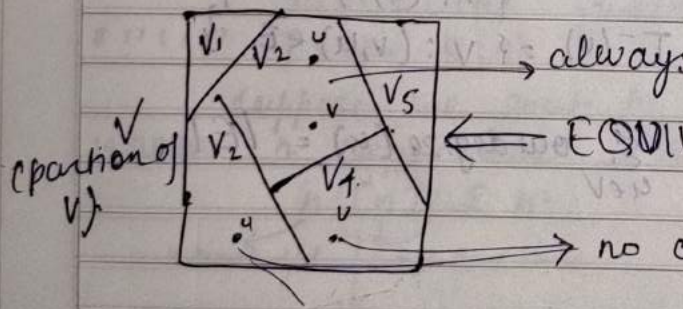
$u \sim v$ is related to

Equivalence Relation:

i) Reflexive $u \sim u$

2) Symmetric $u \sim v \Rightarrow v \sim u$

3) Transitivity $u \sim v$, $v \sim w \Rightarrow (u \sim w)$ → may be walk not path.

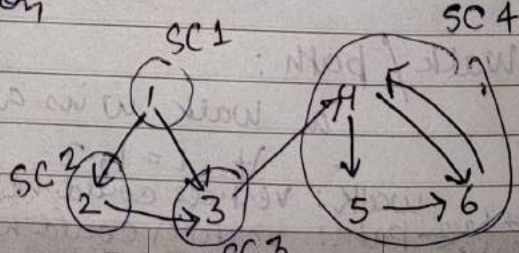


→ always path b/w in set

⟸ EQUIVALANCE CLASS

→ no connection

Directed Graphs: $G(V,E)$



1999

JANUARY

M	T	W	T	F	S	S	WK
			1	2	3		1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL

M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY

M	T	W	T	F	S	S	WK
31					1	2	18
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE

M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES | PHONE CALLS

WEDNESDAY NOVEMBER 17

G is strongly connected if there is a path from every vertex u to every vertex v.

" strongly connected " components

$u \sim v$ if there is a path from u to v & there is a path from v to u

i) reflexive ✓
2) symm. ✓ the equivalence classes define strongly connected components.
3) transitivity ✓

(4) Trees, forest

undirected: connected graph w/o any cycle

(They are minimally connected graphs)
(removing any edge dis connected)

Fact: if G is a tree, n vertices then it has n-1 edges ↑ ⊛

Proof: By induction

$n=1$ $n=1$ edges = 0

our argument [induction: suppose statement ⊛ is true for any tree on n-1 vertices

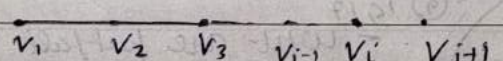
→ Let G be a tree on n vertices

↓

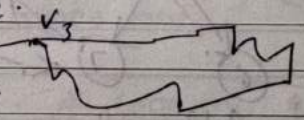
let's remove a leaf → 1st make sure some leaf exist

suppose every vertex has degree ≥ 2. # vertices ≠ ∞

there will be atleast one vertex of d=1



we form a chain $v_1 v_2 \cdots v_i$ ∵ graph is finite some vertex will be repeated → cycle.



⇒ (leaf exist)

G' : n-1 vertices connected

↓ no cycle.

A tree → IH has n-2 edges

⇒ G has n-1 edges

1999

JULY

M	T	W	T	F	S	S	WK
			1	2	3	4	27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST

M	T	W	T	F	S	S	WK
30	31					1	31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER

M	T	W	T	F	S	S	WK
			1	2	3	4	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

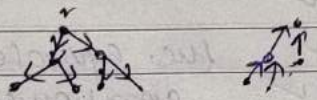
18 THURSDAY NOVEMBER

MESSAGES	PHONE CALLS

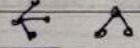
Trees i) connected
ii) No cycle
iii) n-1 edge

(i) (ii) ⇒ (iii) ✓
(i) (iii) ⇒ (ii) } DIY
(ii) (iii) ⇒ (i)

Directed : in tree, out trees



Forest :



arborescence.

How to store a graph.

1) Adjacency Matrix ←
A[i,j] = 1 if i & j have edge
directed = symmetric (store 1/2)
O(1) — $O(n^2)$: storage space complexity
work more about

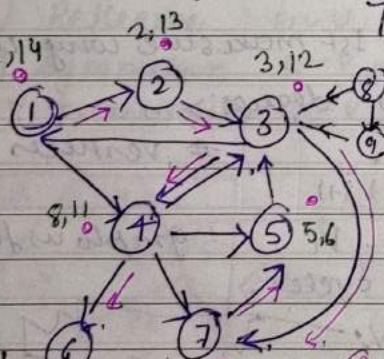
2. Adjacency list
∀ vertex v, we store the list of neighbours of v.
Directed - out
space = $n + \sum_{v \in} \deg(v) = n + 2m = O(n+m)$.

What kind of questn do we answer for a graph.

i) u,v : is an edge (u,v) ?
ii) what are the neighbours.

A: Is G connected? strongly conn? ↙ if not connected!
B: u,v : is there a path from u to v
C: Robust?: if we remove one v, G should not be disconn.
shortest path.
O(1).

Class. 4 19/Aug.



Traversal of or a graph. : starting from s discover the graph
• start from s, keep walking
- what are pitfall ? "dead end" ⇒ no outgoing edge.
- we might stuck in loop ↓ Soln Backtrack.

m/m sort of required

① visited [ ] ← array.
② Parent [ ] ← the preceding vertex but ↓
OR stack : going to a new vertex : push onto
(DFS) Backtrack : pop from the stack.

5
4
3
2
1

comes automatically from RECCURSION

1999

JANUARY

M	T	W	T	F	S	S	WK
			1	2	3		1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL

M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY

M	T	W	T	F	S	S	WK
31				1	2	3	18 19
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE

M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES | PHONE CALLS

FRIDAY NOVEMBER 19

DFS(v) visited[v] = false : DFS(s)

Push : DFS call
Pop : DFS return

```
DFS(v) {
    visited[v] = true;
    for all our neighbours of v
        if visited[u] = false.
            DFS(u).
```

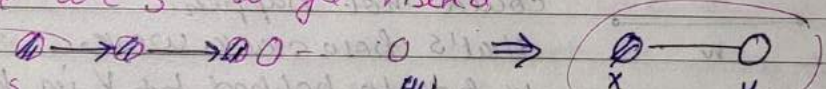
Running Time: 1) for each vertex we c/d DFS(v) at most once.
(bcz visited[v] is set to true now)
2) how much time inside them take.

$$\sum_{v} (1 + \text{degree}(v)) = n + 2m = O(n+m)$$

Correctness: $S = \{v : \text{there is a path from } s \text{ to } v\}$.

i) DFS(u) = this will exactly visit all vertices in S
proof: induction on order of vertices in which they are visited
base case: s is visited
$v_1, v_2 \dots v_i, v_{i+1} \in S$ ← the set of vertices visited by DFS(s)
$s \rightarrow v_1 \rightarrow v_{i+1}$

ii) let $w \in S$ - w get visited.



there must be some edge like this

DFS tree edge.

Suppose w is not visited by DFS(s)
DFS(x) was called.
- consider y — ① DFS(y) gets c/d
② DFS(y) is not c/d.
↳ visited[v] was true. CONTRADICT

How to explore the entire graph:
- start from one another unvisited vertices. (for loop on v; can be used to count connected components)
(this does not affect running time too)

Observation:
- there must be a path from u to v (v_0 to v_r)
- $v_i v_{i+1}$ is an edge.

1999

JULY 7
M T W T F S S WK
1 2 3 4 27
5 6 7 8 9 10 11 28
12 13 14 15 16 17 18 29
19 20 21 22 23 24 25 30
26 27 28 29 30 31 31

AUGUST 8
M T W T F S S WK
30 31 1 31
2 3 4 5 6 7 8 32
9 10 11 12 13 14 15 33
16 17 18 19 20 21 22 34
23 24 25 26 27 28 29 35

SEPTEMBER 9
M T W T F S S WK
1 2 3 4 5 36
6 7 8 9 10 11 12 37
13 14 15 16 17 18 19 38
20 21 22 23 24 25 26 39
27 28 29 30 40

OCTOBER 10
M T W T F S S WK
1 2 3 40
4 5 6 7 8 9 10 41
11 12 13 14 15 16 17 42
18 19 20 21 22 23 24 43
25 26 27 28 29 30 31 44

NOVEMBER 11
M T W T F S S WK
1 2 3 4 5 6 7 45
8 9 10 11 12 13 14 46
15 16 17 18 19 20 21 47
22 23 24 25 26 27 28 48
29 30 49

DECEMBER 12
M T W T F S S WK
1 2 3 4 5 49
7 6 8 9 10 11 12 50
14 13 15 16 17 18 19 51
21 20 22 23 24 25 26 52
28 27 29 30 31 53

20 SATURDAY NOVEMBER

MESSAGES	PHONE CALLS

DFS TREE:

When we call DFS(v); the stack give in a path from s to u

s → u

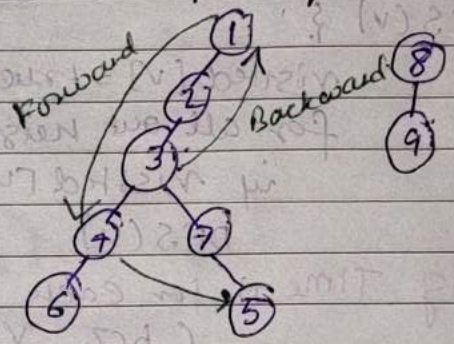
x = u

while (x ≠ s)

x = parent[x]

The edge (parent[u], u) for every u

Sub graph of G



Class 5:

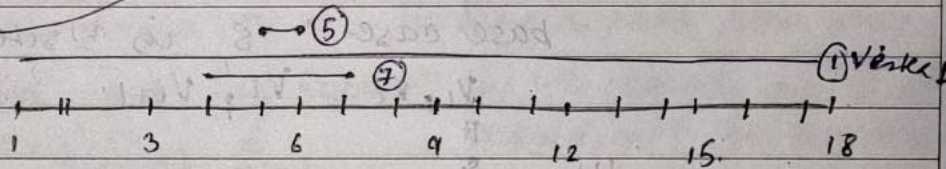
Kleinberg Tardos : Algorithm Design

- Induction
- Contradiction
- Invariance

- Clock → when fn calls happen ↗ Clock ++ (end) ↗ Clock ++

↓ for discovery time

for finished time



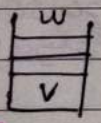
Observations:

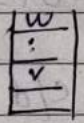
(1) ⇒

could not happen.

that's how stack work.

w has to popped bt v is popped. F[u] < F[w]



(2) ⇒ at a time  Stack is like this

CLAIM: v will be an ancester of w in DFS tree.

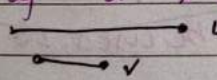
why?

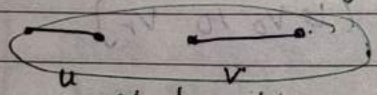
parent[v'] = v ⇒ in the DFS tree v' a child of v.

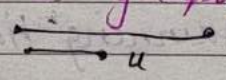
21 SUNDAY

is the converse True? Yes.

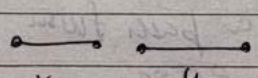
(3) ⇒ if u → v is an edge in graph which is not possible

(1)  Forward edge

(2)  not possible

(3)  v backedge

when we end DFS(u), v have been visited

(4)  Cross edge

1999

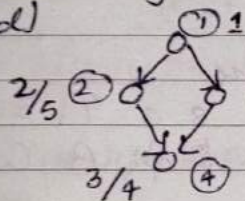
MESSAGES | PHONE CALLS

MONDAY NOVEMBER 22

As special case if G is undirected (i) ≡ (iii) (iii) ⇒ (i) could not happen
all edge must be back edge.

Application:

① Given a graph G, how do we find a cycle?
(undirected) if G has any edge other than T, T = DFS(G).
(directed)



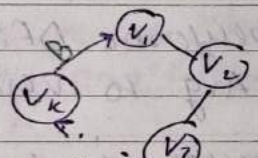
no cycle ∀.

① if there is a backedge ⇒ there is an backedge

is converse true?
if there is a cycle does there has to have a backedge

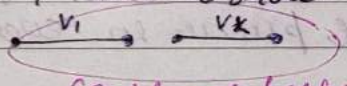
suppose G has a cycle C

There does not mean DFS went like this is cycle.

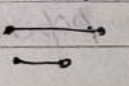


suppose v_1 is the vertex with smallest D[v] value any all vertices in C.

{ need to show v_1 is ancester of v_k }

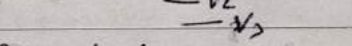


Could not happen



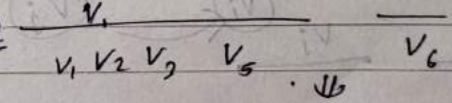
will always happen

proof by contradiction

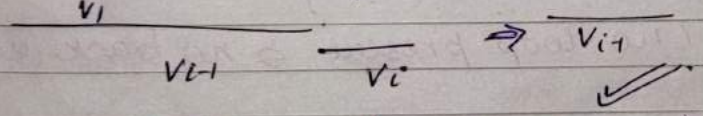


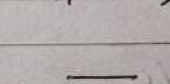
the interval v_2 — v_k ④ are all inside the interval for v_3

suppose ④ is not true ⇒
Let i be the smallest index



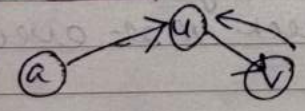
at some point somebody will go out





but v_5 to v_6 has edge.

~~Significance DAG~~ 2 how to check if ∃ backedge while doing DFS



$4 \to u$ $D(v) < D(u)$
and for this time it is not defined

1999

JULY

M	T	W	T	F	S	S	WK
			1	2	3	4	27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST

M	T	W	T	F	S	S	WK
30	31				1	1	31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER

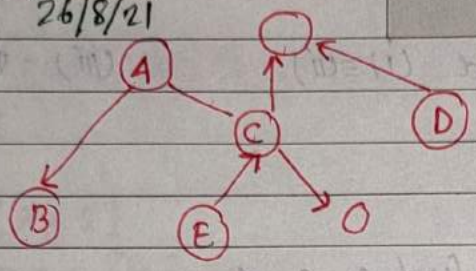
M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

23 TUESDAY NOVEMBER

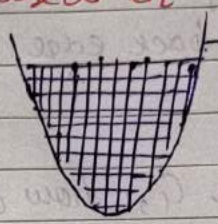
MESSAGES	PHONE CALLS

CLASS-6 26/8/21

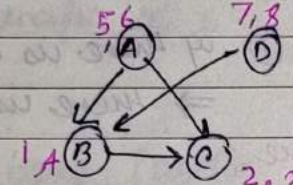
DAG:-



used in - compiler design
- open pit mining (divide core in pts & join it using 4 neighbour & create DAG for max



Topological Sort :- is ordering of vertices in a seq st all edges go from left to right.



8 6 4 3 Finish
D A B C

$V_{\sigma_1}, V_{\sigma_2}, V_{\sigma_3}, \ldots V_{\sigma_n}$ all prop. $\leftarrow V_{\sigma_i}$

Alg:- Average the vertices perform DFS to visit all the vertices ← D[v], F[v] value.. Arg to vertices in dec ord of F[v] value.

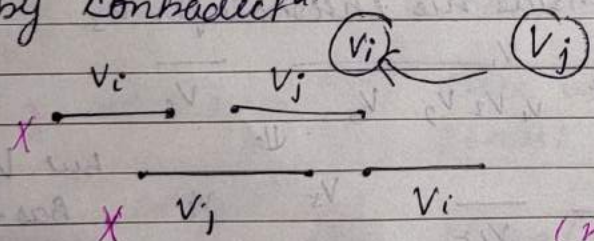
sort- n log n.

We can use stack (put v in list while poping and print in reverse order).

why does it work?

$V_1 \; V_2 \; V_3 \; \ldots \; V_n$ dec ord of $F(v)$ ⇒

by contradict^n.



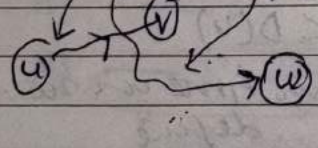
$j > i$ $F[v_i] > F[v_j]$ given

V_i one of v_j

(no loop present ⇒ no back edge)

Strongly connected:-

- Run DFS (v) & check that every vertex is visited
- Remove (G), DFS (v) ; check that every



G | R(G)

1999

JANUARY

M	T	W	T	F	S	S	WK
				1	2	3	1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL

M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY

M	T	W	T	F	S	S	WK
31				1	2	3	18 19 20
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

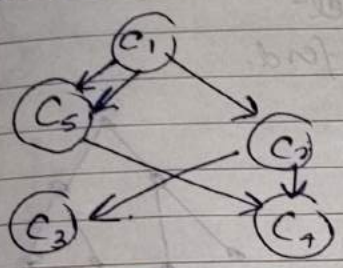
JUNE

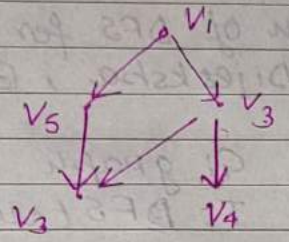
M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES | PHONE CALLS

WEDNESDAY NOVEMBER 24

SCC Assign.





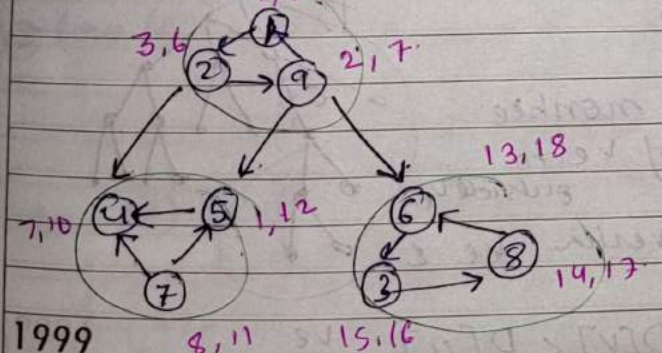
sink → out degree = 0
source → in degree = 0

Kosarajus Algorithm / Kosaraju : suppose we perform DFS from u which is in a sink SCC
↳ congrats ⇒ you got-scc

how do we identify such vertex ???
- Run DFS
- look at the vertex with the highest F[v] value. (this vertex will be in C ← any of the starting node)
(the moment you start with vertex which is not source vertex you can't reach at source (DAG → it is).

since you want to end up in sink vertex (Kosarajus) CC then reverse edge source & sink 8 is reversed in this case.

① G^{rev} run DFS on this
$w_1, w_2 \ldots w_n$ dec ord of F[v] value.
for (i = 1 ... n)
if w_i is not visited // on G.
DFS (w_i) ⇐ see which are the new vertices which are visited



rev sorted of Finish time.
6 8 3 5 7 4 9 2 1
identified sink
next sink
6 8 3 found.
5,7 4 found

1999

JULY 7
M T W T F S S WK
1 2 3 4 27
5 6 7 8 9 10 11 28
12 13 14 15 16 17 18 29
19 20 21 22 23 24 25 30
26 27 28 29 30 31 31

AUGUST 8
M T W T F S S WK
30 31 1 31
2 3 4 5 6 7 8 32
9 10 11 12 13 14 15 33
16 17 18 19 20 21 22 34
23 24 25 26 27 28 29 35

SEPTEMBER 9
M T W T F S S WK
1 2 3 4 5 36
6 7 8 9 10 11 12 37
13 14 15 16 17 18 19 38
20 21 22 23 24 25 26 39
27 28 29 30 40

OCTOBER 10
M T W T F S S WK
1 2 3 40
4 5 6 7 8 9 10 41
11 12 13 14 15 16 17 42
18 19 20 21 22 23 24 43
25 26 27 28 29 30 31 44

NOVEMBER 11
M T W T F S S WK
1 2 3 4 5 6 7 45
8 9 10 11 12 13 14 46
15 16 17 18 19 20 21 47
22 23 24 25 26 27 28 48
29 30 49

DECEMBER 12
M T W T F S S WK
1 2 3 4 5 49
7 6 8 9 10 11 12 50
14 13 15 16 17 18 19 51
21 20 22 23 24 25 26 52
28 27 29 30 31 53

25 THURSDAY NOVEMBER

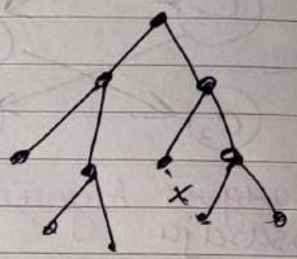
MESSAGES	PHONE CALLS

Class - 7.

- One more application of DFS for undirected
- BFS, connected with Dijekstra, Bellman ford.

DFS for undirected: G: graph
T: DFS tree.

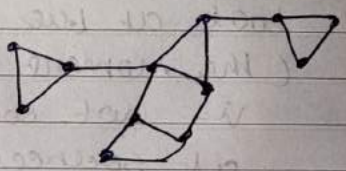
All edges in G-T : should be back edge.



$u \rightarrow v$ $\quad$  $\quad$ could not happen.

=

G(V,E): undirected. we say that an edge $e \in E$ is a cut-edge if removing e disconnect the graph.



how to check if an edge e is a cut-edge? $O(|V|+|E|)$ time using DFS
↳ $O(E)$ if graph is connected.
as $|E| \geq |V| + 1$

How do we find all cut-edge? $O(|E|^2)$.

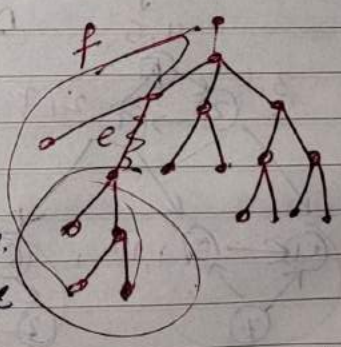
Def: An edge e is a cut edge iff there is no cycle containing it.

Algo:
Run DFS:

Claim: ~~If~~ Any edge $e \in G-T$ can not be a cut-edge

$O(|V||E|)$

Can we do it in linear time?



e is a cut-edge iff there is no nontree edge which goes from subtree of v_e to outside v_e.

e is a cut edge if there is a vertex v_e & $v \notin v_e$ st $u, v \in E$ back edge

$D[v] < D[u]$ v_e

1999

JANUARY								1
M	T	W	T	F	S	S	WK	
			1	2	3	1		
4	5	6	7	8	9	10	2	
11	12	13	14	15	16	17	3	
18	19	20	21	22	23	24	4	
25	26	27	28	29	30	31	5	

FEBRUARY								2
M	T	W	T	F	S	S	WK	
1	2	3	4	5	6	7	6	
8	9	10	11	12	13	14	7	
15	16	17	18	19	20	21	8	
22	23	24	25	26	27	28	9	

MARCH								3
M	T	W	T	F	S	S	WK	
1	2	3	4	5	6	7	10	
8	9	10	11	12	13	14	11	
15	16	17	18	19	20	21	12	
22	23	24	25	26	27	28	13	
29	30	31					14	

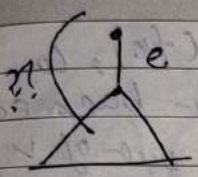
APRIL								4
M	T	W	T	F	S	S	WK	
			1	2	3	4	14	
5	6	7	8	9	10	11	15	
12	13	14	15	16	17	18	16	
19	20	21	22	23	24	25	17	
26	27	28	29	30			18	

MAY								5
M	T	W	T	F	S	S	WK	
31					1	2	18 19	
3	4	5	6	7	8	9	19	
10	11	12	13	14	15	16	20	
17	18	19	20	21	22	23	21	
24	25	26	27	28	29	30	22	

JUNE								6
M	T	W	T	F	S	S	WK	
	1	2	3	4	5	6	23	
7	8	9	10	11	12	13	24	
14	15	16	17	18	19	20	25	
21	22	23	24	25	26	27	26	
28	29	30					27	

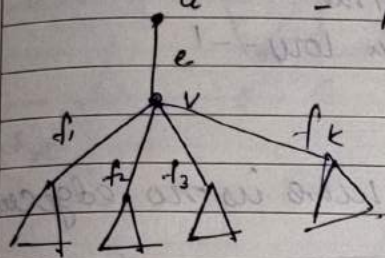
MESSAGES | PHONE CALLS

FRIDAY NOVEMBER 26



linear time

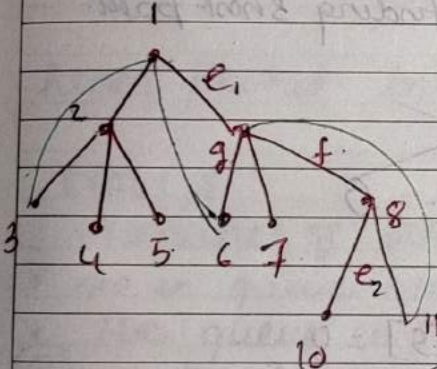
- DFS traversal of the DFS tree & check this for every edge e
- = post order / pre order.



- suppose we know $f_1, \ldots f_k$ which of them are cut edge.
- can we figure out whether e is a cut-edge or not

Is e cut edge?

low [f] : ~~if u v~~ it is the ~~highest~~ such that there is a back edge (u,v) $v \in V_f$.
highest pt out of subtree.



low $[e_1]$ = 1.

low (f) =

suppose know
low $[f_1]$, low $[f_2]$, ... low $[f_k]$
can we figure out low [e] ?
D [ ] and the back edge from v.

highest away

Let w_i be the one with min D[w] value $w_j \equiv V$! low [e] = -1

(a) low (e) = w_i if $w_i \neq V$
-1 if $w_i = V$ }.

w_i : look at all the edge f_i where low $[f_i] = -1$ among them pick the vertex w/ min D[ ] value.

low (e) }
{ if e is an leaf edge.
(u, v)
low [e] = highest back edge going out of v

1999 months

JULY 7

M	T	W	T	F	S	S	WK
		1	2	3	4		27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST 8

M	T	W	T	F	S	S	WK
30	31					1	31/36
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER 9

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER 10

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER 11

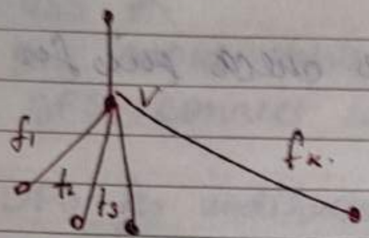
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER 12

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

27 SATURDAY NOVEMBER

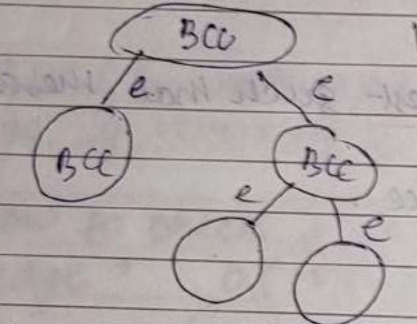
MESSAGES	PHONE CALLS



compute low [f1] .. compute (fk), lo:
low [i] = ⊙ highest back edge going out of v

O (|V| + |E|) time
o/1 all edges with low - - 1

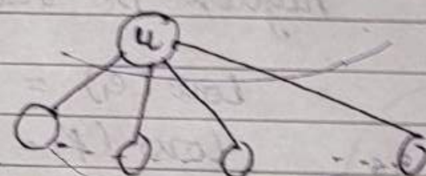
Bi - Connected:
A graph is said to be biconnected if there is no edge cut



BFS

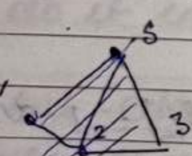
DFS

DFS is not good for finding short path.



BFS: tell All your neighbours
Queue ← enque Dequeue.
Enqueue (s) visited [S] = true. level [S] = 0
At each step:
repeat till queue is empty { V = Dequeue (Q)
for all neighbours w of v ⌒ visited [w] = false.
Enqueue (w). , visited [w] = true, level [w] = level [v] + 1.

28 SUNDAY



for v = 1 .. n
if visited [v] = false
BFS (v)

P [w] = V.
parent info

1999

JANUARY

M	T	W	T	F	S	S	WK
				1	2	3	1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL

M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY

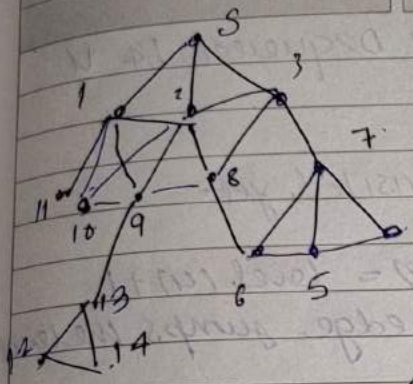
M	T	W	T	F	S	S	WK
31					1	2	18
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE

M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	22
7	8	9	10	11	12	13	23
14	15	16	17	18	19	20	24
21	22	23	24	25	26	27	25
28	29	30					26

MESSAGES | PHONE CALLS

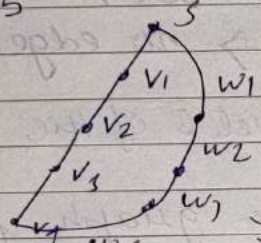
MONDAY
NOVEMBER 29



how do we prove that BFS tree is a shortest path.

Any $(u, v) \in E$ $\quad$ // level (u) − level (v)// ≤ 1

5 edge S to V_5 in the BFS tree



suppose there is a shorter path from S to V_5: 4 edge //

⇒ any one of them is jumping 1 level which is not possible

Hence proved. that we have shortest path.

Class 8:

properties of BFS:

0	1	1	1	2	2	2	2
S	x	x	3	9	10	11	8

- no in queue always be like $x, x+1, x+1, \ldots, x+2, x+2, \ldots$
- the queue will always has following properties

(P)
- the levels of this vertices are in ascending order from front to rear
- levels of any two vertices on queue by at most 1.

something will always true for always ⇒ Induction

Prove by induction in this case.

i) satisfied at begining

ii) suppose it is satisfied at some step then 'v' is also satisfied at next step.

i) | S |
0

ii) Deque(v) enque some of the neighbour of v.

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

or:

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

1999

JULY

M	T	W	T	F	S	S	WK
		1	2	3	4		27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST

M	T	W	T	F	S	S	WK
30	31					1	31/35
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER

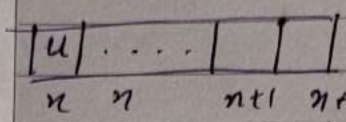
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

30 TUESDAY NOVEMBER

MESSAGES



PHONE CALLS

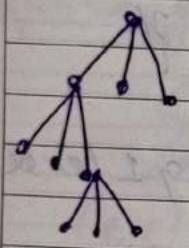
(*) Suppose $(u,v) \in E$ let us say u is dequeued b4 u. (at some time t).

where is v ? i) v is in Q at time.
ii) v has not been visited yet.

level(v) = level(u) or level(u)+1

level(v) = level(u)+1

at most 1 difference is possible ⇒ no edge jumps the level.



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

$S - V_1 - V_2 - \cdots - V$

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

$S \quad w_1 \quad w_2 \cdots V$

$v_0 \quad \leq 1 \quad \leq 2 \quad w_k$

level(w_i) $\leq i$ level(v) $\leq k \Rightarrow k \geq l$

there is no path P of size less than k.

Directed Graph:

neighbours becomes out-neighbours.

$(u,v) \Rightarrow u \longrightarrow v$

10 1.

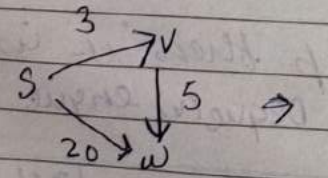
level(v) $\leq$ level(u) + 1

prove it yourself.

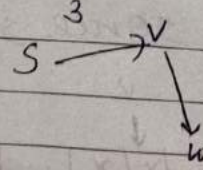
Shortest path for weighted graph: (Directed Graph)

each edge has a weight we.

eg.



⇒



let's take $\forall w_e > 0$

Dijkstra's Algo.

MESSAGES | PHONE CALLS

WEDNESDAY 01 DECEMBER

idea

$\forall e$ $w_e = 1 \rightarrow$ BFS

e — 10

⇓ → subdivide e with we edges & nodes & can now solve using BFS

1 1 1 1 10 time.

depend upon ~~length~~ wt of edge $w_e \in [0, L]$ — $O(|E|L)$

but dijkstra is doing it implicitly.

- visit the graph in BFS like manner
- Dij's Algo will also visit vertices in order of increasing distance from S.

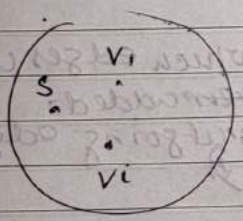
$S, v_1, v_2, v_3, \ldots v_k$ ← vertices arranged in ↑ order of distance fr S.

$0 \quad d_1$

S	v_1	v_2	v_i	v_{i+1}
0	d_1	d_2	d_i	?

$S \xrightarrow{2} v_1$

$S \longrightarrow$ go out from the edge leaving this set



let us consider the shortest path S to v_{i+1}

$S \longrightarrow \longrightarrow \longrightarrow \cdots \longrightarrow v_j \rightarrow v_{i+1}$ (d_j)

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

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

let $S \leftarrow \{s\}$ $D[s] = 0$ visited[v] =

while (S is not all) {

consider all edges $e^{(u,v)}$ in $\delta^+(S)$ $\{u \in S, v \notin S\}$

$\delta^+(S)$

$x_e = D[u] + w(e)$.

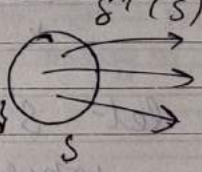
pick the edge with min x_e value e^*

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

visited[v] = false

$O(nm)$

1992



JULY

M	T	W	T	F	S	S	WK
			1	2	3	4	27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST

M	T	W	T	F	S	S	WK
30	31					1	31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER

M	T	W	T	F	S	S	WK
			1	2	3		40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

02 THURSDAY DECEMBER

MESSAGES | PHONE CALLS

Proof of correctness.

After i iterations of the while loop.

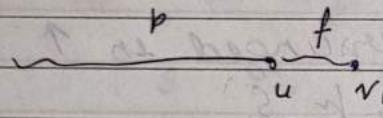
$S = \{s, v_1, \ldots v_i\}$ & $D[v]$ will be the shortest path distance from s to v for all $v \in S$.

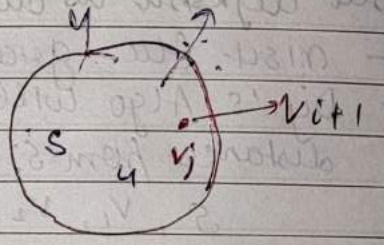
Base case $S = \{s\}$. $D[s] = 0$

Induction $S = \{s, v_1 \ldots v_i\}$ $D[v_1], \ldots D[v_i]$.

Claim: $v = v_{i+1}$ & $x_e = D[v]$.

let P be shortest path from s to v_{i+1}



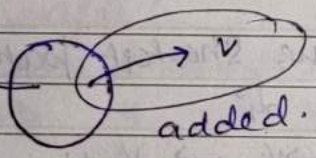


length of s to v path: $x_f = D[u] + l(f) = D[v_{i+1}]$.

$x_{e'} = D[v_j] + l(e')$

$x_e \leq x_f \Rightarrow v = v_{i+1}$ $x_e = D[v_{i+1}]$

Can we improve from here ←



⇒ which incoming to edges to v will be removed.

which edges will be added. (outgoing edges from v)

→ we can use heaps. ? problem will be delete.

→ we can use AVL tree. → decrease value of $e \in S[s]$

→ can use fibonacci heaps.

$\sum_v \deg(v) \log n \Rightarrow O(|E| \log |V|)$ $(m + n \log n)$

let $S \leftarrow \{s\}$ $d(v) \rightarrow \infty$

while ( ) { }

// for every edge $e(u,v)$ in $\delta^+(s)$

$d[v] = D[u] + w(e)$

} same as previous code.

1999

JANUARY 1: M T W T F S S WK; 1 2 3 1; 4 5 6 7 8 9 10 2; 11 12 13 14 15 16 17 3; 18 19 20 21 22 23 24 4; 25 26 27 28 29 30 31 5

FEBRUARY 2: M T W T F S S WK; 1 2 3 4 5 6 7 6; 8 9 10 11 12 13 14 7; 15 16 17 18 19 20 21 8; 22 23 24 25 26 27 28 9

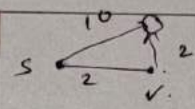
MARCH 3: M T W T F S S WK; 1 2 3 4 5 6 7 10; 8 9 10 11 12 13 14 11; 15 16 17 18 19 20 21 12; 22 23 24 25 26 27 28 13; 29 30 31 14

APRIL 4: M T W T F S S WK; 1 2 3 4 14; 5 6 7 8 9 10 11 15; 12 13 14 15 16 17 18 16; 19 20 21 22 23 24 25 17; 26 27 28 29 30 18

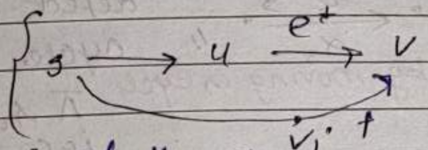
MAY 5: M T W T F S S WK; 31 1 2 18(21); 3 4 5 6 7 8 9 19; 10 11 12 13 14 15 16 20; 17 18 19 20 21 22 23 21; 24 25 26 27 28 29 30 22

JUNE 6: M T W T F S S WK; 1 2 3 4 5 6 23; 7 8 9 10 11 12 13 24; 14 15 16 17 18 19 20 25; 21 22 23 24 25 26 27 26; 28 29 30 27

MESSAGES | PHONE CALLS



pick the vertex $v \in S$ w/ min $d[v]$ value.
Add v to S $D[v] = d(v)$.
}



$D(u) + w(e^*) = x_e$
$D[v_j] + w(f) = x(f)$ } this can't happen.

~~moral of the story~~.
now store the D[V] on heap not Xe values
just get the min.

Class 9 6/9/21

What Dijkstra do ? ~~min~~ $d(V) = \min_{e \in u,v;\ u \in S;\ v \notin S} [w_e + D[u]]$

we want to know min d(V).
- take all edges coming to v from S. & choose the min

D[S] = 0 initially. $d[v] = \infty$ $v \neq S$
repeat {
let v be a vertex visited [v] = false & d[v] is min
visited [U] = true D[u] = d[u]
for all edges e: (u → v where visited [v] = false
d[v] = min [d[v], we + D[u]])
P[V] = u.
en.
}

heap

fibbonacci heap (m + n log n)
heap AVL m log n

what happen if w_e is -ve ?? # -path

Q) $w_e = -1$ ∀ edges ; Longest path min -|P| max |P|

s ⟶ v NP hard problem

1999

JULY

M	T	W	T	F	S	S	WK
			1	2	3	4	27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST

M	T	W	T	F	S	S	WK
30	31					1	31/36
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

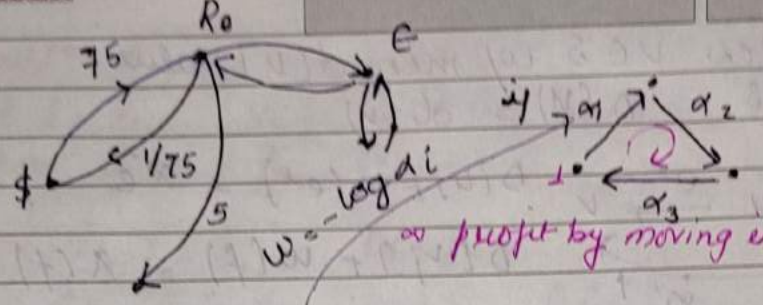
DECEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
6	7	8	9	10	11	12	50
13	14	15	16	17	18	19	51
20	21	22	23	24	25	26	52
27	28	29	30	31			53

04 SATURDAY DECEMBER

MESSAGES	PHONE CALLS

ii) Currencies



if $a_1 a_2 a_3 > 1$ we want to detect if ∃ a cycle s.t $\prod_{i \in C} a_i > 1$

∞ profit by moving in cycl.

$$\sum_{i \in C} \log(a_i) > 0 \text{ or } \sum_{i \in C} -\log(a_i) < 0$$

s-t shortest path

Shortest s-t walk: → most algo tried to solve it.

most algo try to do this

if -ve edge present then length of shortest s-t path walk would be < length of shortest s-t C

-ve cycles.

As long as, ∄ no negative cycle shortest walk ≡ shortest path

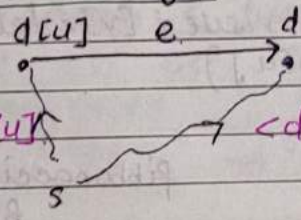
there are polynomial time algo to find the shortest st path

Bellman Ford: $O(mn)$ ≫ $O(m \log n)$ ⇒ dijkstra

maintain $d[v]$ = estimate of shortest path from s to v.

$D[v]$ = length of " " " " "

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



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

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

5 SUNDAY

repeat:

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

$d[v] > d[u] + w_e$

update

$d[v] = d[u] + w_e$

1999

JANUARY								FEBRUARY								MARCH								APRIL								MAY								JUNE							
M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK
				1	2	3	1								6								10				1	2	3	4	14						1	2	18			1	2	3	4	5	6	23
4	5	6	7	8	9	10	2	1	2	3	4	5	6	7	7	1	2	3	4	5	6	7	11	5	6	7	8	9	10	11	15	3	4	5	6	7	8	9	19	7	8	9	10	11	12	13	24
11	12	13	14	15	16	17	3	8	9	10	11	12	13	14	8	8	9	10	11	12	13	14	12	12	13	14	15	16	17	18	16	10	11	12	13	14	15	16	20	14	15	16	17	18	19	20	25
18	19	20	21	22	23	24	4	15	16	17	18	19	20	21	9	15	16	17	18	19	20	21	13	19	20	21	22	23	24	25	17	17	18	19	20	21	22	23	21	21	22	23	24	25	26	27	26
25	26	27	28	29	30	31	5	22	23	24	25	26	27	28	10	22	23	24	25	26	27	28	14	26	27	28	29	30			18	24	25	26	27	28	29	30	22	28	29	30							27

MESSAGES | PHONE CALLS

MONDAY DECEMBER 06

until no edge can be found.

Why will it terminate ??

$D[v] \le u$

$O(nU)$ # steps

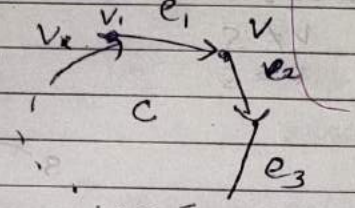
⇒ this procedure will run into ∞ loop if ∃ a -ve cycle

Proof: suppose (n), suppose it terminate

e_i = it must be the case

this must be the case.

$d[v_{i+1}] \le we_i + d[v_i]$



K inequalities

add them all

$d[v_2] \le we_1 + d[v_1]$

$d[v_3] \le we_2 + d[v_2]$

$d[v_i] \le we_3 + d[v_3]$

$0 \le we_1 + we_2 + we_3$ } contradiction with starting

writing Algo in a diff way.

$e_1, e_2 \dots e_m$

~~for~~ $i = 1 \dots n-1$

~~for~~ $j = 1 \dots m$

suppose $e_j = (e_{ij}, v_j)$

if $d[v_j] > d[u_{ij}] + we_j$

$d[v_j] = d[u_{ij}] + we_j$

if ∃ an edge $e(u \to v)$

$d[v] > d[u] + we$ then deduce -ve cycle.

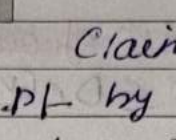
Claim ∀ vertices $d[w] \ge D[w]$

P.T. by induction on # steps

base case. $d[v] = \infty$ ✓

when ⓐ happen is it still true?

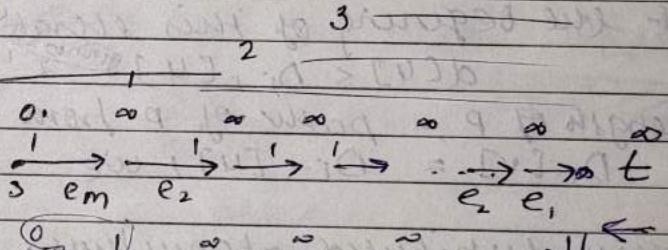
we relaxing $d[v]$



by IH P has length $\le d[u]$.

$(\because d[u] \ge D(u))$

$D[v] \le d[u] + we$??



you almost ~~me~~ need to do it- $O(m^2)$

$:=$

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

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

1999

07 TUESDAY DECEMBER

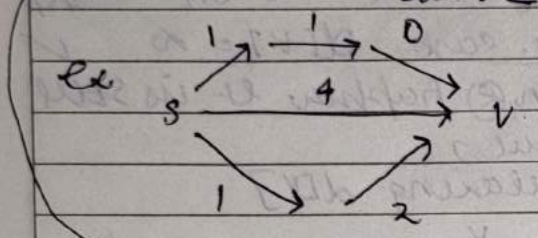
MESSAGES

PHONE CALLS

Claim: After i iterat^{n} of outer for loop

$d[v] \leq D_i[v]$ $\quad i = n-1$ $\quad d[v] = D_{n-1}[v]$

$D''[v]$: atmost i edge



$D_0[v] = \infty$

$D_1[v] = 4$

$D_2[v] = 3$

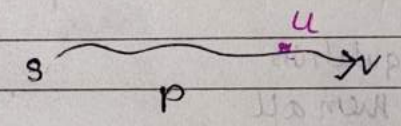
$D_3[v] = 2$

Proof by induct^{n} of i

i) $\rightarrow$ $i = 0$? $\quad d[v] = \infty \quad d[s] = 0$

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

ii) $\rightarrow$ suppose it is true for $i-1$



$\leftarrow$ $d[u] \leq D_{i-1}[v]$

i'th

$d[v] \leq D_i[v]$ (*) $\leftarrow$

let P be the shortest $s-v$ path of length atmost i

at the beginning of this iteration $d[u] \leq D_{i-1}[u]$ given to us when we look at this e during the for loop

Length of P = path of P from $s-u$ + edge e.

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

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

very distributed algorithm. every edge can update its own.

Class 10 9/sep/21.

Greedy Algorithm: $\rightarrow$ computational problem: min/max some quantity

optimal solution: a sol^{n} which achieve this min/max

Coin changing problem.

values / $c_1, c_2, c_3 \ldots c_k$

denomination

∞ supply

Goal: find an exact change for u units as few coin as possible.

1999

MESSAGES	PHONE CALLS

WEDNESDAY 08 DECEMBER

each step do something that seems right at that point- pick the biggest coin $< M$ C_n & has, $M - C_n$

eg $C_1 = 1, C_2 = 7, C_3 = 10$

$M = 14$ 2 coins of 7

10, 1, 1, 1, 1 5 coins acc to greedy. LOL.

what if $C_1 | C_2$, $C_2 | C_3$, ... $C_{r-1} | C_r$! then greedy is optimal.

how do we prove that a greedy is correct ??

i) the 1st step is correct & agrees w/ the algo on 1st step.

ii) use induction

Optimal soln: $S_1 \mid S_1\ S_2 \ldots S_k$ — After taking the 1st step the problem becomes the same type of problem.

$S_2 \mid O_2$? →

1st step agrees on this ↑ optimal soln O where the 1st step is also S_1. P_1: problem where we have already taken the 1st step.

$O_2, O_3 \ldots$

O, $S_2 \ldots S_k$ is only better than $O_1\ O_2\ O_3 \ldots$ M

coin changing problem: $C_1 \ldots C_k$ $C_1 | C_2, C_2 | C_3 \ldots$

M: pick the largest coin $C_i < M$, $M - C_i$

coin chosen

$a_1 \geq a_2 \geq \ldots a_n$ } i) $a_1 = O_1$ think about M- 1st step

$O_1 \geq O_2 \geq \ldots O_k$ } ii) Both $a_2 \ldots a_n$ & $O_2 \ldots O_k$ are feasible change $M - a_1$

rejection is here

Claim: greedy Alg is optimal. Induction on M greedy optimal substructure problem.

proof- by induction $n - 1 \leq k - 1$

$\Rightarrow n \leq k$

n = # coins in optimal soln

k = # coin w/ greedy

EXCHANGE ARGO ??

1999

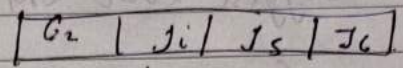
JULY								AUGUST								SEPTEMBER								OCTOBER								NOVEMBER								DECEMBER							
M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK	M	T	W	T	F	S	S	WK
			1	2	3	4	27							1	31 35			1	2	3	4	5	36					1	2	3	40						1	2	3	4	5	49				1	2	3	4	5	49
5	6	7	8	9	10	11	28	2	3	4	5	6	7	8	32	6	7	8	9	10	11	12	37	4	5	6	7	8	9	10	41	1	2	3	4	5	6	7	45	7	6	8	9	10	11	12	50
12	13	14	15	16	17	18	29	9	10	11	12	13	14	15	33	13	14	15	16	17	18	19	38	11	12	13	14	15	16	17	42	8	9	10	11	12	13	14	46	14	13	15	16	17	18	19	51
19	20	21	22	23	24	25	30	16	17	18	19	20	21	22	34	20	21	22	23	24	25	26	39	18	19	20	21	22	23	24	43	15	16	17	18	19	20	21	47	21	20	22	23	24	25	26	52
26	27	28	29	30	31		31	23	24	25	26	27	28	29	35	27	28	29	30				40	25	26	27	28	29	30	31	44	22	23	24	25	26	27	28	48	28	27	29	30	31			53
																																29	30						49								

09 THURSDAY DECEMBER

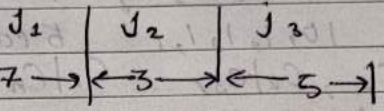
MESSAGES	PHONE CALLS

Weighted completion time Scheduling :

job j_1 j_2 - - j_n
size p_1 p_2 - - - p_n
weight w_1 w_2 w_n



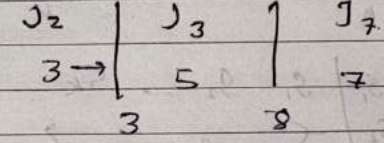
$j_1, j_2, j_5 \equiv 7\ 3\ 5.$



schedule : an order in which to process the jobs

C_j : completion time of job

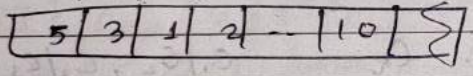
Goal : min $\sum_{i=1}^{n} C_{ji} \cdot W_{ji}$



Greedy Algo : $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq - - \geq \frac{w_n}{p_n}$

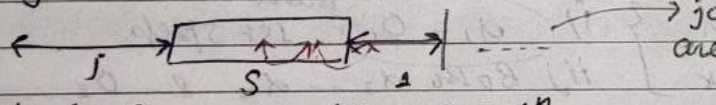
sort jobs in this order & execute it in this order

1) There is an opt algo which also processes job 1 1st



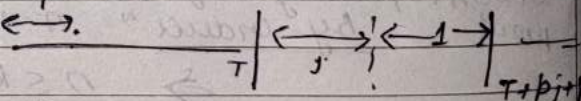
1, 2, 3 - - 10

exchange argument : consider an opt soln O



Modify opt to a new optimal soln

swap j & 1 to more toward greedy soln, but make sure Goal will be going to better



trying = Bf the swap T_i

$(T + P_j) w_j + (T + P_j + P_1) w_1$

After the swap

$(T + P_1) w_1 + (T + P_j + P_1) w_j \leq ?$

1999

JANUARY

M	T	W	T	F	S	S	WK
			1	2	3		1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL

M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY

M	T	W	T	F	S	S	WK
31					1	2	1823
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE

M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES | PHONE CALLS

FRIDAY 10 DECEMBER

$P_i w_j \leq P_j w_i$ (everything else changes)

$\frac{w_1}{P_1} \geq \frac{w_j}{P_j}$ (true) do it in recursive manner

swap till i is at start

ii) suppose the greedy algo is optimal when there $< n$ jobs suppose we have n jobs $\frac{w_1}{P_1} \geq \cdots \frac{w_n}{P_n}$.

by step (i) there is an opt sol^n $O_1, \ldots O_n$ where $O_1 = 1$

1	2, 3 ...	n	Steps
O_1	O_2 ...	O_n	

by induction hypothesis: I.H

weighted comp time $(2, 3, \ldots n) \leq$ weight comp $(O_2, \ldots O_n)$ (same greedy Algo on smaller job)

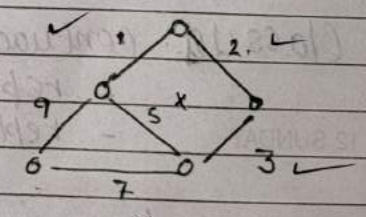
weighted comp $(1 \ldots n) = wc(2, 3 \ldots n) + [w_2 P_1 + \cdots + w_n P_1]$.

weight comp $(O_1 \ldots O_n) = wc(O_2, \ldots O_n) + w_2 P_1 + \cdots + w_n P_1$

optimal sol^n greedy sol^w is better or equal to optimal (since it can't be better $\Rightarrow$ it is same.

Minimum Spanning Tree:

$G = (V, E)$ e: we connected



Goal: pick a spanning tree of min total wt

greedy Alg $we_1 \leq we_2 \leq we_m$.

$T \leftarrow \phi$

repeat {

if adding e_i to T does not create a cycle pick e_i & add it to T

1999

JULY

M	T	W	T	F	S	S	WK
		1	2	3	4		27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST

M	T	W	T	F	S	S	WK
30	31				1		31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

DECEMBER

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

11 SATURDAY DECEMBER

MESSAGES

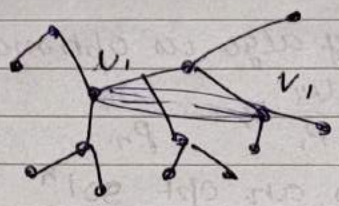
PHONE CALLS

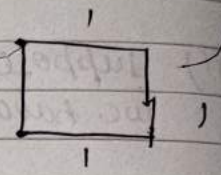
why is this optimal: pick edges

i) Let us say greedy $g_1\ g_2 \dots g_{n-1}$: G $w_{g_1} \le w_{g_2} \le \dots w_{g_{n-1}}$

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

O also contain g_1 ?? not sure but if an optimal soln which has g_1 in O





~~Ado~~ let O does not g_1

$g_1 (u, v)$

Add G_1 to the soln O; creates a cycle. Remove any other edge from this cycle

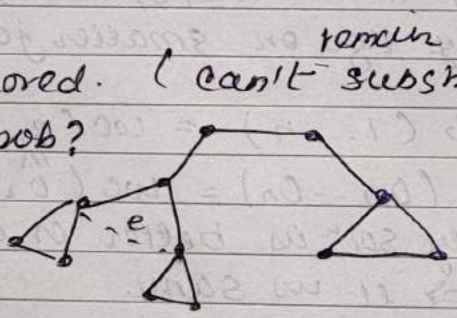
$\Rightarrow$ the cost will always go down or remains same

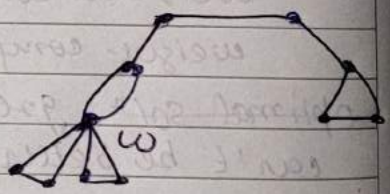
{ modify something in O to show that greedy's 1st step can be incorporated }

ii) let's say that e is removed. (can't remain substitute property

Now what is remaing prob?

that is of same kind





contract e.

Class 10: contract an edge $e = (u, v)$

- replace u, v to a new vertex w.
- replace all edges (u, x) or (v, x) by (w, x) [except for ~~all~~ all edges b/w

12 SUNDAY

Suppose there is a spanning tree T in G which contain e

the T_e: tree obtain 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.

1999

JANUARY

M	T	W	T	F	S	S	WK
			1	2	3		1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL

M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY

M	T	W	T	F	S	S	WK
31				1	2		18/23
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE

M	T	W	T	F	S	S	WK
		1	2	3	4	5	6
7	8	9	10	11	12	13	23
14	15	16	17	18	19	20	24
21	22	23	24	25	26	27	25
28	29	30					26
							27

MESSAGES	PHONE CALLS

MONDAY
DECEMBER 13

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

greedy : $g_1 \ldots g_{n-1}$ $\quad g_1 = o_1 = e$

opt $\quad o_1 \ldots o_{n-1}$

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

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

by induct^n hypothesis $\quad \underline{\text{cost}(g_2 \ldots g_{n-1}) \leq \text{cost}(o_2 \ldots o_{n-1})}$

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

$\Rightarrow$ cost (greedy) on $G \leq$ cost (opt) on G

example: what if we started w/ prim's Algo
start by taking min wt edge & go on.

huffman encoding :

A A B AA C AA BC AAA ← given a text / string

↓ encode as binary string

00 00 01 00 00 10

2n

A: 00
B: 01
C: 10

Σ : alphabet
eg. $\Sigma = \{A, B, C\}$

variable length encoading

A: 0 $\quad$ 00 10 0 0 11 $\quad$ much smaller.
B: 10
C: 11

take another case.

A: 0	A C AC	B D C
B: 01	0 11 0 11	0 11 0 11
C: 11		
D: 10		

} → prefix free property $l_1 \ldots l_k$
encoading (l_1), encoding (l_2) ..
none of these is the prefix of the

1999

JULY 7

M	T	W	T	F	S	S	WK
			1	2	3	4	27
5	6	7	8	9	10	11	28
12	13	14	15	16	17	18	29
19	20	21	22	23	24	25	30
26	27	28	29	30	31		31

AUGUST 8

M	T	W	T	F	S	S	WK
30	31					1	31
2	3	4	5	6	7	8	32
9	10	11	12	13	14	15	33
16	17	18	19	20	21	22	34
23	24	25	26	27	28	29	35

SEPTEMBER 9

M	T	W	T	F	S	S	WK
		1	2	3	4	5	36
6	7	8	9	10	11	12	37
13	14	15	16	17	18	19	38
20	21	22	23	24	25	26	39
27	28	29	30				40

OCTOBER 10

M	T	W	T	F	S	S	WK
				1	2	3	40
4	5	6	7	8	9	10	41
11	12	13	14	15	16	17	42
18	19	20	21	22	23	24	43
25	26	27	28	29	30	31	44

NOVEMBER 11

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	45
8	9	10	11	12	13	14	46
15	16	17	18	19	20	21	47
22	23	24	25	26	27	28	48
29	30						49

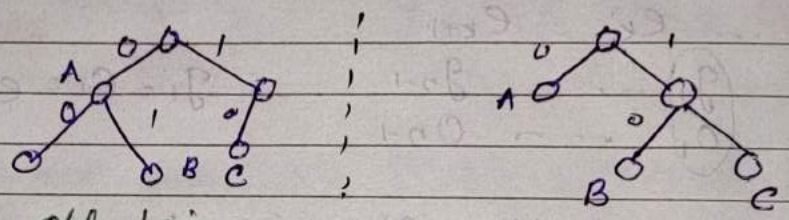
DECEMBER 12

M	T	W	T	F	S	S	WK
		1	2	3	4	5	49
7	6	8	9	10	11	12	50
14	13	15	16	17	18	19	51
21	20	22	23	24	25	26	52
28	27	29	30	31			53

MESSAGES | PHONE CALLS

this encoding can be seen as a tree st all labels should be on leafs (distinct)

A: 0
B: 01



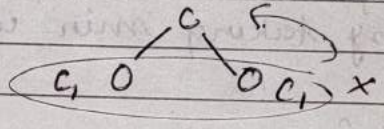
this data structure is c/d. trie.

problem: given an string, find a variable length encoding (trie) with min length of the encoading

A	B	C	D
10	50	30	70

try to give max length encoding to words which are less frequent.

C_1 depth(C_1) · fC_1
C_2 depth(C_2) · fC_2.



Algo: let C_1, C_2 be the two letters with freq (fc_1, fc_2).

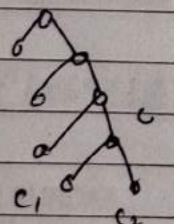
if only 2 letters 0,1 ↓

replace C_1, C_2 by a new letter C 1+01 1

$fC_1 + fC_2 = fc$

solve the smaller problem. let us say the encoding C is σ (σ=0110)

encoding for C_1 C_2 are $\sigma 0$, $\sigma 1$.



$\leq l+1$ $\leq l+1$

earlier $fc_1 \cdot l_1 + fc_2 l_2$

Now $(fC_1 + fC_2) \cdot l$

Diff $fC_1 + fC_2$ does not depend on the tree

Ex:

	01	00	10	11						
	A	B	C	D		0 E	10 C	11 D		0 E	1 F
C:	5	1	1	1		2	1	1		2	2

A D

1) there is an opt sol^n which makes with the ols on the last step
2) Apply induct^n.

1999

JANUARY
M	T	W	T	F	S	S	WK
			1	2	3		1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH
M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

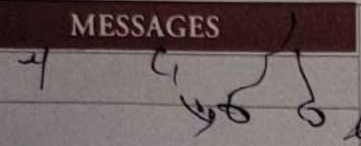
APRIL
M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY
M	T	W	T	F	S	S	WK
31					1	2	18 23
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

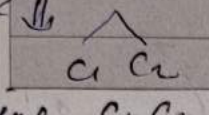
JUNE
M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES | PHONE CALLS

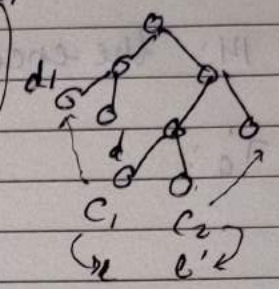
WEDNESDAY 15 DECEMBER



$C_2 \Rightarrow$ then I am going to see that new sol^n is only better/equal to opt



Greedy: 1st combine $c_1 c_2$ into a new letter e
we want to prove that there is an opt sol^n.
~~where to prove that there is an opt sol^n.~~
the leaves c_1 c_2 have a common parent.



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

$e \leftarrow d$ } $e \leftarrow d_1$
$c_1 \leftarrow d_1$ } $c_1 \leftarrow d$.

$f_e d + f_{c_1} d_1 \geq f_e d_1 + f_{c_1} d$.

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

$\Leftrightarrow (f_{c_1} - f_{c_2})(d - d_1) \geq 0$
$\geq 0 \quad \geq 0$

Class 12: 16/sep.

i) We showed that there is optimal sol^n T^* where c_1 c_2 have a common parent.

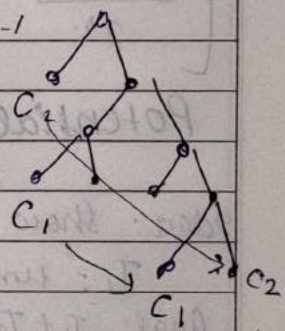
ii) induct^n proof:

Assume that the greedy alg is optimal where there $< k-1$ letters.

k letters $a_1 \dots a_k$.

a) the 1st step greedy alg.

combine $c_1 c_2 \to c$.



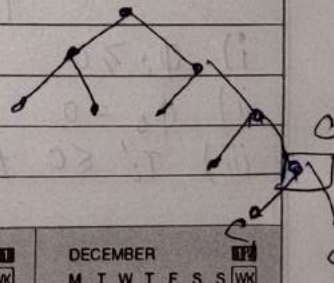
By (i) there is an opt sol^n T^* where c_1, c_2 have a common parent

$\Sigma' = (\Sigma - \{c_1, c_2\}) \cup \{c\}$: k_1 letters.

By induct^n hypothesis the greedy edge is opt for Σ'

T': T^* w/ $c_1 c_2$ removed & the common parent labelled c.

then $(T^*)'$ is also sol^n for Σ'



1999 opt & greedy agrees on 1st steps.

$\mathcal{E} \to \mathcal{E}' \to$ IH for $\mathcal{E}'$ $T_G \to \mathcal{E}$.

16 THURSDAY DECEMBER

MESSAGES | PHONE CALLS

T_G'

By IH: the encoding length of greedy on $\mathcal{E}' \leq$ length of $(T_G)'$

T_G': T_G: greedy soln for $\mathcal{E}$

$(T_G)'$: greedy for $\mathcal{E}'$

=

Next Topic: Amortized complexity: total time / k.

$\to$ [A] $T(n)$.

$i_1, i_2, \ldots i_k \to A$. ($|n|$, n) → this is worst case running time

$O(kT(n))$

example: Bit increment:

2. 101 110 11 ← 11

+1

101 111 00

bit operations = 1 + # consecutive 1's at the end

$\left\{ \frac{\sum_{i=1}^{2^n} T_i = ?}{2^n} \leq 2 \right\}$

Potential function: ϕ_i : non-ve number at the beginning of the ith step.

Idea: Show that when the alg. takes lot of time potential drops]

T_i: time taken by the alg at the i^{th} step

Goal: $T_1 + T_2 + \ldots T_n \leq C(n)$: one way of showing is that $T_i \leq C(n)/n$ ||

modified time at step i

$T_i' = T_i + \Delta\phi_i = T_i + \phi_{i+1} - \phi_i$ ——— (2)

i) $\phi_i \geq 0$

ii) $\phi_0 = 0$

iii) $T_i' \leq C$ for each i

$\sum_{i=1}^{n} T_i' \leq C \cdot n$

1999

JANUARY

M	T	W	T	F	S	S	WK
			1	2	3		1
4	5	6	7	8	9	10	2
11	12	13	14	15	16	17	3
18	19	20	21	22	23	24	4
25	26	27	28	29	30	31	5

FEBRUARY

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	6
8	9	10	11	12	13	14	7
15	16	17	18	19	20	21	8
22	23	24	25	26	27	28	9

MARCH

M	T	W	T	F	S	S	WK
1	2	3	4	5	6	7	10
8	9	10	11	12	13	14	11
15	16	17	18	19	20	21	12
22	23	24	25	26	27	28	13
29	30	31					14

APRIL

M	T	W	T	F	S	S	WK
			1	2	3	4	14
5	6	7	8	9	10	11	15
12	13	14	15	16	17	18	16
19	20	21	22	23	24	25	17
26	27	28	29	30			18

MAY

M	T	W	T	F	S	S	WK
31					1	2	18
3	4	5	6	7	8	9	19
10	11	12	13	14	15	16	20
17	18	19	20	21	22	23	21
24	25	26	27	28	29	30	22

JUNE

M	T	W	T	F	S	S	WK
	1	2	3	4	5	6	23
7	8	9	10	11	12	13	24
14	15	16	17	18	19	20	25
21	22	23	24	25	26	27	26
28	29	30					27

MESSAGES	PHONE CALLS

$T_1' = T_1 + \phi_2 - \phi_1$

$T_2' = T_2 + \phi_3 - \phi$

$T_n' = T_n + \phi_{n+1} - \phi_n$

$\Rightarrow T_1' + \cdots T_n' = T_1 + \cdots + T_n + \underbrace{\phi_{n-1} - \phi_1}_{\geq 0}$ (with $\phi_1 \to 0$)

? $T_1 + \cdots T_n \leq C n.$

x_i : number of the begening of i^{th} step ($\equiv i$ in binary)

ϕ_i = # 1's in the binary expansion of i ($\equiv$ # 1's in x_i)

claim: $T_i' \leq 2 \Rightarrow T_1 + \cdots T_n \leq 2n$ as well.

$\Rightarrow \sum T_i \leq 2n$

$\phi_1 = 0$

$\phi_i \geq 0$

Pf: x_i (i)

```
  110110 -- 10
           + 1
  ------------
  110110 -- 11
```

$\phi_{i+1} - \phi_i = 1$

$T_i = 1.$

ii).

```
  110110 -- 011111   (l ones)
                 1
  ----------------
  ...      100000
```

$\phi_{i+1} - \phi_i = 1 - l$

$T_i = l + 1$

$T_i' = 2 \qquad T_1 + T_2 + \cdots T_n = 2n$

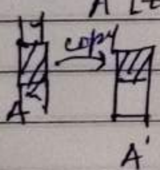
how can we define ϕ for Random function. ?? not easy.

Expandable array.

Adding x at the end of the array

$A[t+i] = x$

Amortized time: n.



what is the good stratergy to decide size of A'

ins(x') {
  if A is full
  copy A into an array of twice the size
  then add x at the end
  else add x at the end