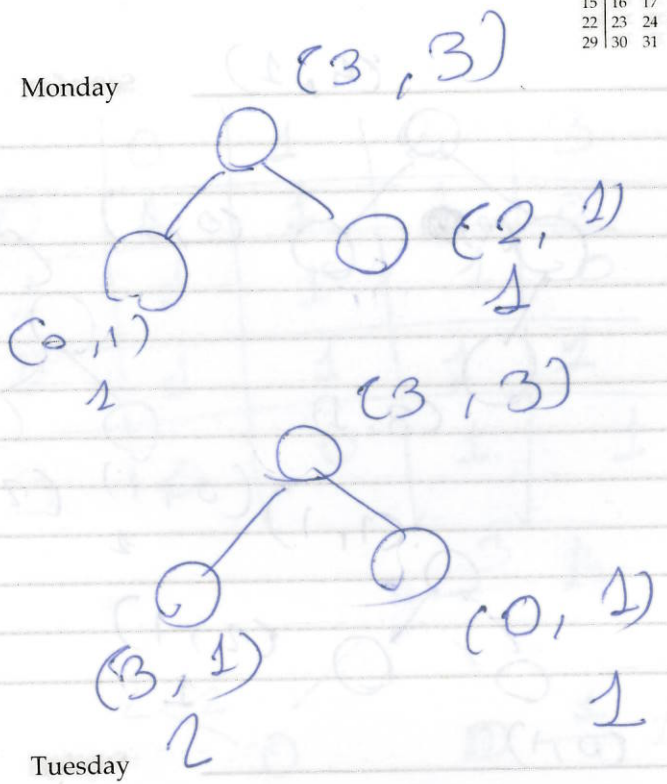
DECEMBER 2013

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

27 Monday



28 Tuesday

FEBRUARY 2014

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

Wednesday 29

MIT-2

→ 006 is pre-requisite

⇒ 004

- divide & conquer
- optimization
  - greedy
  - DP
- Network flow
- Intractibility
  - approximaton algorithms
- Advance topics
  - distributed algo
  - cryptography

Thursday 30

Theme of today's lecture

- Very similar problem can have very different compleaty
- P, NP & NP-complete

JANUARY 2014

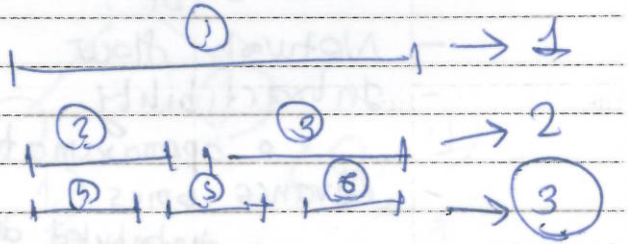
DECEMBER 2013

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

31 Friday

Interval Scheduling

- Resource & Requests
  $s(i) < f(i)$
- Two request i & j are compatible if they don't overlap



- Optimization problem:
  select compatible subset of requests / intervals of maximum size.

claim: we can solve this problem using a greedy algorithm

- A greedy algorithm is a myopic algorithm that processes one input at a time without look ahead

JANUARY 2014

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

FEBRUARY

Monthly Planner

Schedule your appointments, events and important dates

MARCH 2014

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

1 Sat	2 Sun	3 Mon	4 Tue
5 Wed	6 Thu	7 Fri	8 Sat
9 Sun	10 Mon	11 Tue	12 Wed
13 Thu	14 Fri	15 Sat	16 Sun
17 Mon	18 Tue	19 Wed	20 Thu
21 Fri	22 Sat	23 Sun	24 Mon
25 Tue	26 Wed	27 Thu	28 Fri

Action Plan for Previous Month

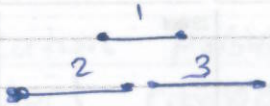
Greedy Interval Scheduling

1. Use a simple rule to select a request i
2. Reject all request that are incompatible with i
3. Go back to step 1 until all request are processed

Review

Rule

① select one with earliest finish time
② select one with smallest path.



③ For each request find no. of incompatible request
- select the one with minimum incompatible

Self Analysis

→ We need to prove our greedy rule mathematically.

Claim: Given a list of interval L, greedy algorithm with earlist finishish time produces k^* interval, where k^* is maximum.

Proof
- Induction on k^*
- Base case:-
  $k^* = 1$ Any interval works
- Suppose claim holds for k^* and we are given list of intervals whose optimal schedule has $k^* + 1$ intervals

$S^*[1, 2, \ldots k^*+1]$
$= \langle S(j_1), f(j_1) \rangle, \ldots$

$\langle S(j_{k+1}), f(j_{k+1}) \rangle$

$S[1, \ldots k] = \langle S(i_1), f(i_1) \rangle, \ldots$
$\langle S(i_k), f(i_k) \rangle$

above k and k^* are not comparable.

$f(i_1) \leq f(j_1)$

* Weighted Interval Scheduling

- 2^n will make it exponential
- We ~~want~~ to can use DP
- Subproblem :
  - There can be many possible way to create sub problem, all valid

$$R^x = \{ \text{request}_j \in R \mid S(j) \geq x \}$$

$$x = f(i)$$

$R^{f(i)}$ → request later than $f(i)$

- n = no. of request
- # subproblem = n
- time / subproblem = n
- Complexity : $O(n^2)$

DP guessing

- Try each request i as possible first request

$$opt(R) = \max_{1 \leq i \leq n} \left(w_i + opt(R^{f(i)}) \right)$$

Complexity : $O(N^2)$

→ There is smarter DP formulation $O(N \log N)$

* Non identical machines

$T = \{ T_1, \ldots T_m \}$

weight = 1 for all request

$Q(i) \in T$ ~~else~~

→ set of machine where i can run on

→ NP complete problem

JANUARY 2014

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

1 Saturday

Lec-2 Divide & Conquer

→ Paradgm

- Given a problem of size n, Divide it into "a" subproblems of size n/b
- $a \geq 1$, $b > 1$
- Solve each sub-problem recursively
- Combine solutions of subproblems into overall solution (merge)

2 Sunday $T(n) = a \cdot T(n/b) + [\text{work for merge}]$

- Master theorem for solving above recurrence

* Convex Hull

- Given n points in a plane
  $S = \{(x_i, y_i) \mid i = 1, 2, \cdots n\}$

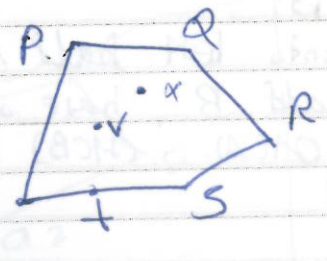
  assume no two have the same x cord, no two have same y, cord, no three in a line

MARCH 2014

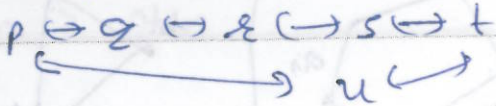
S	M	T	W	T	F	S
30	31					1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

3 Monday

- Convex hull is smallest convex polygon containing all points in S CH(S)



- sequence of points on the boundary in clockwise orders as doubly linked list



4 Tuesday

⇒ Brute-Force

- Draw line, if all other points are one side of the line it is a segment
- Complexity :- n points
  - $O(n^2)$ segment
  - Complexity of test $O(N)$
  - Total $O(n^3)$

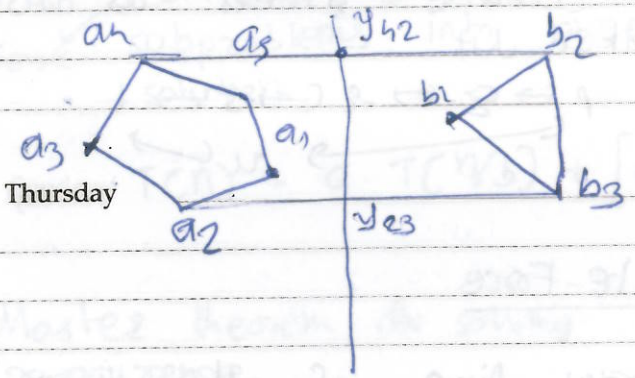
JANUARY 2014

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

5 Wednesday

- Sort points by x coodinates (once for all)
- For input set S^1
  - Divide into left half A & right half B, base on x-cood.
  - solve O(CA), (CB)
  - combine

How to merge?

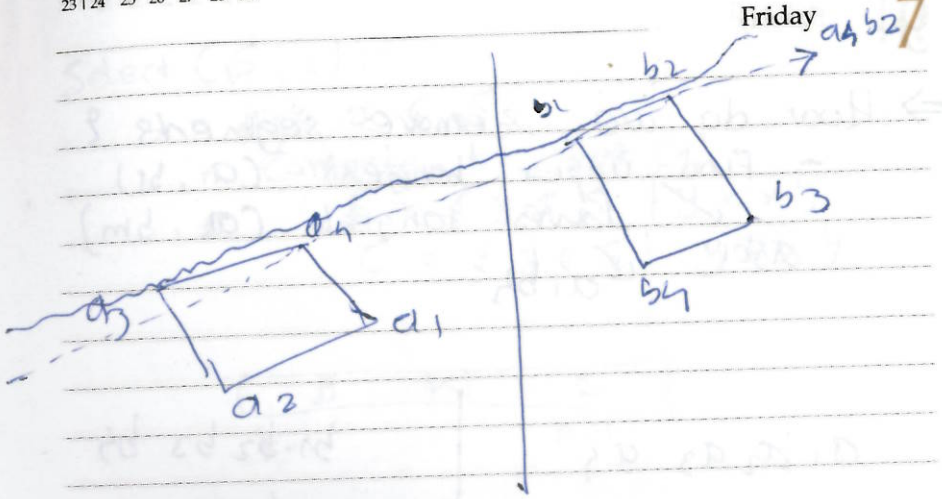


6 Thursday

- pair of segment from each pair and test like brute-force
- a_4, b_2 is upper tangent

→ Obvious merge algo :-
- $O(n^3)$ pairs of points
- Here it is not $O(n^3)$ because we will find maximum point on the

Friday 7



→ So O(1) is not clear.
→ $y_{31} > y_{42}$

Saturday 8

⇒ 2 finger algorithm

- We are starting with points nearest to divider
- Complexity : O(N)

⇒ $T(n) = 2 \cdot T(n/2) + O(n)$
$= O(n \log n)$

JANUARY 2014

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

9 Sunday

⇒ How do we remove segments ?
- Find upper tangent (a_i, b_j)
  ✓ lower tangent (a_k, b_m)
  $a_3 b_1$ ✓ $a_1 b_4$

$a_1\ a_2\ a_3\ a_4$ | $b_1\ b_2\ b_3\ b_4$

First let $a_1 \rightarrow b_j \rightarrow$ go down b list till you see $b_m \rightarrow a_k \rightarrow$ continue till you return a_1

10 Monday

- Complexity O(N)

* Median Finding

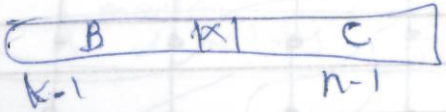
- We want complexity better than $O(n \log n)$
- Given set n numbers, define rank(x) as, count of number $\leq x$
- rank $\left\lceil \frac{n+1}{2} \right\rceil$ is lower median rank $\left\lceil \frac{n+1}{2} \right\rceil$ upper median

MARCH 2014

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

Tuesday 11

Select (S, i) :-
- pick $x \in S$
- Compute k = rank(x)
  $B = \{ y \in S \mid y < x \}$
  $C = \{ y \in S \mid y > x \}$



if k = i :-
  return x
elif k > i :- return select(B, i)
else k < i :- return (C, i-k)

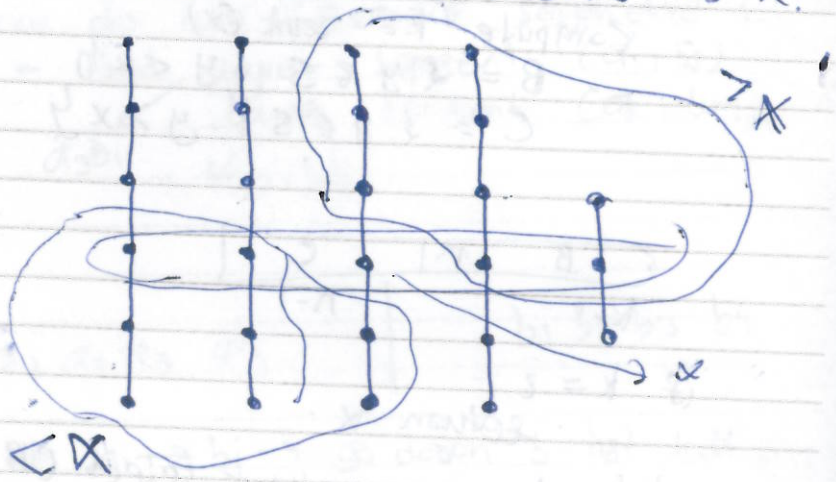
Wednesday 12

⇒ we need to specify how we pick up x.
- worst case can be $O(n^2)$

⇒ Pick x cleverly
- Arrange S into columns of size 5 ($\lceil \frac{n}{5} \rceil$ columns),
- sort each column big elements in bp. 5 elements ⇒ linear

13 Thursday

- Find "median of medians" as x.



14 Friday

⇒ How many elements are guaranteed to be > x.

- Half of $\lceil n/5 \rceil$ group contribute at least 3 element ≥ x, except for 1 group with less than 5 element & 1 group that contains x

= At least $3\left(\lceil \frac{n}{10} \rceil - 2\right)$ > x

" < x

15 Saturday

Recurrence:

$$T(n) = \begin{cases} O(1) & : \text{for } n \leq 140 \\ T\left(\lceil \frac{n}{5} \rceil\right) + T\left(\lceil \frac{3n}{10} \rceil + 6\right) + O(n) & \end{cases}$$

$\lceil \frac{n}{5} \rceil$ → median of medians

$\lceil \frac{3n}{10} \rceil + 6$ → discarding $\frac{3n}{10} - 6$ elements

$O(n)$ → sorting at all columns

→ Above runs in $O(n)$ time

16 Sunday

Lec-9 Augmentation Range Trees

Idea: take "off the shelf" data structure and modify to store extra information

- Mostly we will use balanced trees

JANUARY 2014

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

17 Monday

* Easy tree Augmentation

- goal :- to store f(subtree rooted @ x) at each node x in x.f

- Suppose x.f can be computed in O(1) time from x, children and children.f

- if modify set S of nodes then cost O(# ancestors of S) to update f fields.

18 Tuesday

- which would be O(log n) in
  - AVL trees
  - 2-3 trees

* Order statistic trees

- ADT :-
  - insert(x)
  - delete(x)
  - successor(x)
  - rank(x) :- index of x in sorted order
  - select(i) :- give me key of rank i

- All above in log(n)

MARCH 2014

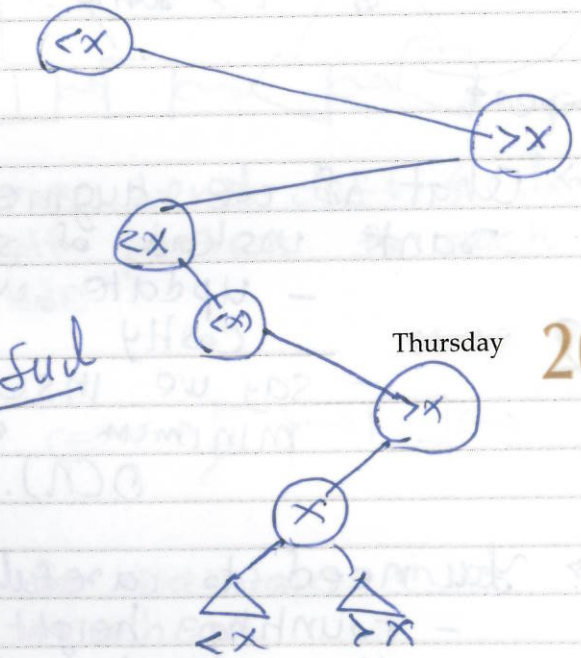
S	M	T	W	T	F	S
30	31					1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

Wednesday 19

- use easy tree augmentation with f(sub-tree) = # nodes in the subtree

$x.f = 1 + \text{sum}(c.f \text{ for } c \text{ in } x.\text{children})$

size



Wonderful

Thursday 20

rank(x)

- rank = x.left.size
- walk up from x to root.
  - whenever we go left ($x \to x'$) rank += x'.left.size + 1

21 Friday

- select (i):
  - x = root
  - rank = x.left.size + 1
  - if i = rank : return x
    if i < rank : x = x.left
    if i > rank : x = x.right
    i = i - rank

repeat

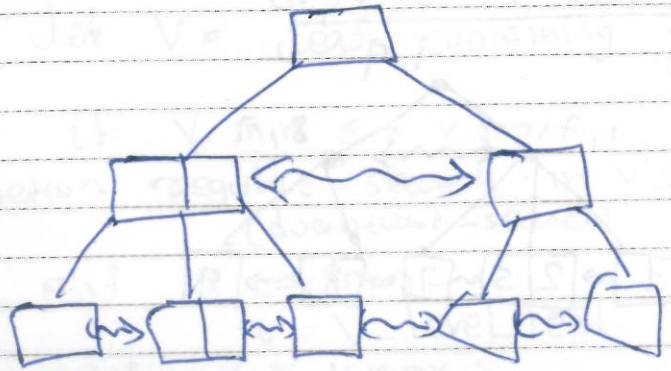
→ What if we augment with rank instead of size?
- update would be costly
- say we insert new minimum element O(n).

22 Saturday

→ You need to careful
- counting height is easy then depth when it comes rotating

23 Sunday

* Level linked 2-3 trees



Idea: store level-left & level right pointers for each node

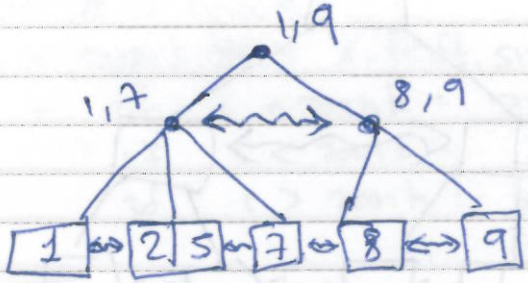
24 Monday

- easy to maintain in O(1) overhead

* Finger search property
  - search (x from y)
    O(log | rank(x) - rank(y) |) time

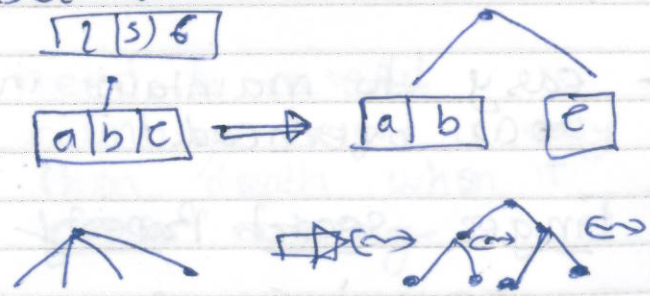
25 Tuesday

* Store data n leaves



Search
- augment :- min and max key in sub-tree
- O(log n) search

26 Wednesday

Insert
- insert 6
- 

Thursday 27

Search (x from y)

- let V = leaf containing y
- if V.min ≤ x V.max :-
  return regular-search in V's subtree (downward-search)
  elif x < V.min :-
    V = V.level_left
  elif x > V.max :-
    V = V.level-right
  V = V.parent

Friday 28

→ Above achieves finger-search bound
- At iteration :-
  - I am at height k
  - I skip $\Omega(2^k)$
- Say diff is d, between ranks of x and y
- Downward search cost log k same as rest of search
- Total complexity log(k).

Geet

FEBRUARY 2014

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

MARCH

Monthly Planner

Schedule your appointments, events and important dates

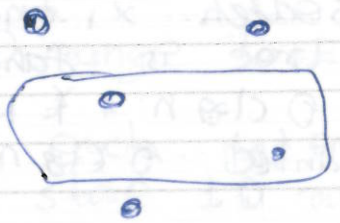
APRIL 2014

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

1 Sat	2 Sun	3 Mon	4 Tue
5 Wed	6 Thu	7 Fri	8 Sat
9 Sun	10 Mon	11 Tue	12 Wed
13 Thu	14 Fri	15 Sat	16 Sun
17 Mon	18 Tue	19 Wed	20 Thu
21 Fri	22 Sat	23 Sun	24 Mon
25 Tue	26 Wed	27 Thu	28 Fri
29 Sat	30 Sun	31 Mon	

Action Plan for Previous Month

* Range Trees (Orthogonal range search)



Review

2D

- Query would be rectangle box, give me points inside them $(\log n)^2$
- In 3D query would be 3D-cube $(\log n)^3$

goal: preprocess n-points in d-dimension to support a query
- given axis align box
find:-
① # points in box
② k points in box → O(k) for output

Self Analysis

Achieve $O(\log^d n + \|output\|)$

1-D

- binary search x_1 and x_2
- segment tree is fancy
- BST :- $O(\log n + k \log n)$
- level linked :- $O(\log n + k)$

1-D range tree

- perfect BST

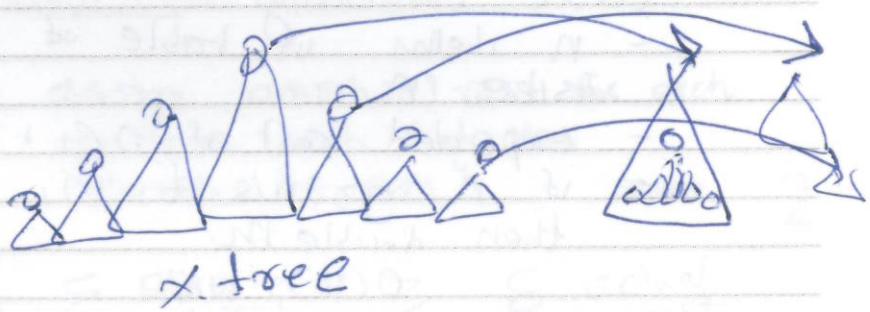
① ⇒ range-query([a,b])
- search(a)
- search(b)
- trim common prefix

- its normal BST, elements not stored on leaves
- it will return nodes of rooted sub-tree
- $O(\log n)$

2D range tree

(These things are useful in database query filter)

- 1D range tree of all points by x
- for each node v in x-tree store 1D range tree by y on all points of v's roots subtree.



x-tree

- pre-processing, hence data shifting is okay
- $O(\log^2 n)$

#. Space complexity
- each key leaves in $\log n$ subtrees (ancestor)
- $O(n \log^{d-1} n)$

- We are losing log factor for each dimension you add.

Week 5

Amortization

→ Analysis of data structures
- specially when DS is used for algorithm
- For example, dykstra uses heap

Recall
→ table doubling
- n items in table of size m
- expected cost of $O(1 + m/n)$
- if n grows to m, then double m
  $O(m)$ work
- you double only $O(\log m)$ time
- total cost of n insertions
  $= O(2^0 + 2^1 + \dots + 2^{\log n})$
  $= O(n)$
- "amortized cost per operation is constant $O(1)$"

Saturday 1

* Aggregate method
→ total cost of k operation / k
= amortized cost per operation

* Amortized Bounds
- assign amortize cost for each operation such that you "preserve the sum"

Sunday 2

$\sum_{op}$ amortized cost $\geq$ $\sum_{op}$ actual cost

e.g.
2-3 trees achieve
- $O(1)$ worst case per create-empty-tree
- $O(\log^* n)$ amortized per insert
- ~~$\emptyset$~~ amortized per delete.
  $O(0)$
  → You can not delete more than you have inserted

3 Monday

- In actual case delete will be $O(\log n)$
-

total cost $= O(C + i \log n^* + d \log n^*)$

$= O(C + i \log n^*)^{d \le i} + d \cdot O$

4 Tuesday

* Accounting Method

- Bank account
- Allow an operation to store credit in bank account ($C \ge 0$)
- allow operations to pay for time using credit in bank.
- balance ≥ 0

Wednesday 5

Claim:
- $O(\log n)$ per insert
- 0 per delete, amortized
- put 1 coin with $O(\log n)$ per insert
- delete, consumes 1 coin

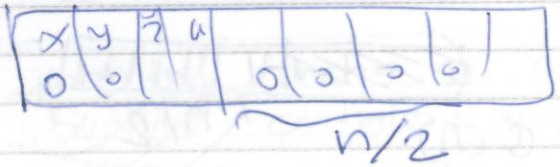
amortized cost = actual cost + deposits − withdrawl

- invariant:
  1 coin @ $\log i$ for $i = 1, 2, \ldots, n$

Thursday 6

Table doubling

- when insert, add coin on that item with $C = O(1)$



- when double, last $n/2$ items have coins $n/2$

⇒ amortize cost $= O(n) - c \cdot n/2$
$= 0$ if we set c large

FEBRUARY 2014

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

7 Friday

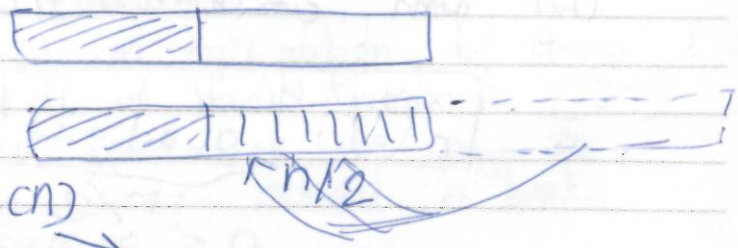
* Charging Method

- allow operations to charge cost retroactively to past (not future)

- amortized cost = actual cost − total charge to past + total charge in future

8 Saturday

⇒ Table Doubling



O(n)

- charge doubling to inserts since last doubling (n/2)
  ⇒ O(1) to each

APRIL 2014

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

Sunday 9

⇒ Table doubling & halving m = O(n)

- 100% full → double
- 25% full → half
  → anything less than 50 will work

- After double/halve, 50% full

→ charge halving to m ≥ 4 deletion since last resize

Monday 10

- charge doubling to ≥ m/2 insertions since last resize

* Potential Method

- define potential function Φ mapping data structure configuration to non-negative integer
- amortized cost = actual cost + ΔΦ

FEBRUARY 2014

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

11 Tuesday

$\Delta\Phi$ = Φ(after op) − Φ(before operation)

$\sum$ amortized cost = $\sum$ actual costs + Φ(end) − Φ(beginning)

⇒ pay Φ(beginning) at beginning
- should be O(1) or Φ

12 Wednesday

* Binary Counter

- 0 0 1 1 0 1 0 1 1 1
→ INCR 0 0 1 1 0 1 1 0 0 0

→ Only constant no of bits changes in amortized analysis.

- increment cost = O(1 + # trailing 1's)

APRIL 2014

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

Thursday 13

Φ = # 1 bits

→ If there are t trailing bits, increment destroy t 1's create one 1
amortized cost = (1 + t) − (t) + (1)
= 2

— O(1 + t) − ct + 1.c = 2

* 2-3 tree

⇒ Insert in 2-3 trees:-

Friday 14

MARCH 2014

FEBRUARY 2014

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

15 Saturday

Lec-10 ~~Advanced~~ DP

- longest palindrom sub-sequence
  - sub-sequence ⇒ non contiguous

16 Sunday

APRIL 2014

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

2014 MARCH

Monday 17

Lec-6 Randomization

Matrix Multiply & Quick Sort

- Analysis becomes important in randomized algorithms

Defination of randomized algorithm

Algorithm that generates a random number & GJ1, PJ and makes decision based on it

Tuesday 18

- On the same input, on different executions
  1. algo may run for distesent no. of steps
  2. produce different outputs

→ Probably correct - monte carlo | Probably fast - Las Vegas

→ Probably correct & probably fast - Atlantic city

MARCH 2014

FEBRUARY 2014

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

Task is to verify matrix product, not calculate

square matrice $n \times n$

19 Wednesday

* Matrix Product
  $C = A \cdot B$
- Simple algorithm : $O(n^3)$ multiplications
- Strassen : Multiply two 2×2 matrices using 7 multiplication instead of 8 (2.81)
  $O(n^{\log_2 7}) = O(n^{2.81})$
- Copper Smith Winograd : $O(n^{2.376})$

20 Thursday

⇒ We want to get $O(n^2)$ algorithm

If $A \times B = C$ then
prob [output = yes] = 1

If $A \times B \neq C$ then
prob [output = yes] $\leq \frac{1}{2}$

- You can bring-down probability by multiple execution.
  prob $\leq \frac{1}{2^k}$ ⇒ $O(k \cdot n^2)$

APRIL 2014

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

21 Friday

→ entries $\in \{0,1\}$ mod 2

⇒ Freivald's algorithm

- choose random binary vector $\in [1, \ldots n]$
  such that $P_r[r_i = 1] = 1/2$
  independently $i = 1, \ldots n$

If $A(Br) = Cr$ output YES
else output NO

22 Saturday

- How many matrix vector product ? → 3

→ If $AB = C$
then $A(Br) = (AB)r = Cr$
so there are no chance of false ~~negative~~ positive.

23 Sunday

Analysing correctness when $AB \neq C$

claim:- if $AB \neq C$, then
Prob $[AB\xi \neq C\xi] \geq 1/2$

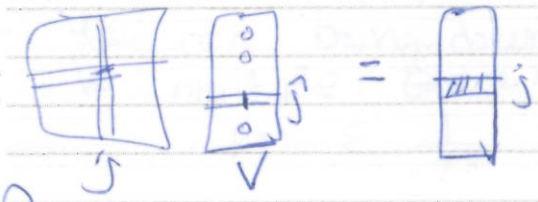
- Let $D = AB - C$, our hypothesis is that $D \neq 0$

- We need to show that there are many ξ such that $D\xi \neq 0$

24 Monday specifically Prob $[D\xi \neq 0] \geq 1/2$ for randomly chosen ξ

→ Case: $D\xi = 0$, $D \neq 0$

$D = AB - C \neq 0 \Rightarrow \exists\ i,j$ s.t $d_{ij} \neq 0$



$\Rightarrow DV \neq 0$
$\Rightarrow DV_j \neq 0$
$\rightarrow d_{ij} \neq 0$

D

Tuesday 25

- Take any ξ that can be chose by our algo.

such that $D\xi = 0$

$\xi' = \xi + V$ (mod 2 arithmetic)

→ vectors

$\rightarrow D\xi' = D(\xi + V)$
$= D\xi + DV$
$= 0 + DV$
$\neq 0$

→ For any given ξ → bad
s.t. $D\xi = 0$ there is

Wednesday 26

→ ξ & ξ' is 1 to 1 if $\xi' = \xi + V$
$= \xi'' + V$

then $\xi = \xi''$ because
i.e. mod 2.

→ discover ξ' s.t. $D\xi' = 0$ → ξ & ξ' is 1, 1
→ No. of ξ' for which $D\xi' \neq 0$
$\geq$ No. of ξ for which $D\xi = 0$

$P[D\xi \neq 0] \geq 1/2$.

27 Thursday

* Quick-sort

- Divide & Conquer algorithm
- In place :- no auxilary space
- all the work is in divide space

⇒ 3 different Variants :-
Basic

⇒ n-element array A

Divide :- ① pick a pivot element x in A

② Partition the array in sub-array

L	E	G
$<x$	x	$>x$

28 Friday

Conquer :-

① Reursively sort sub-array L & G

⇒ CLRS page : 171 for in-place partitioning.

Saturday 29

① Basic Quick Sort

→ pivot $x = A[1]$ or $A[n]$
- partition give x $O(n)$ time

→ Worst case analysis :-
- For sorted or reverse sorted case
- One side L or G has n-1 elements and other (1)
- $T(n) = T(0) + T(n-1) + O(n)$ ← divide step
$= T(n-1) + O(n)$
$= O(n^2)$

Sunday 30

- If we shuffle input at the begining on average it becomes $O(n \log n)$

② Intelligent pivot selection

- gurantee balance L & G using median select. that runs in $O(n)$ time.

$T(n) = 2T(n/2) + O(n) + O(n)$

Not good for performance ← recursive median selection ; divide partition

MARCH 2014

FEBRUARY 2014

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

31 Monday

(3) Randomized Quicksort

- Las vegas algorithm
- x is chosen random from array A
- expected time $O(n \log n)$ for all input arrays A
- CLRS - page 181 & page 184

⇒ variant quick sort

Paranoid Quick sort

Repeat
- Choose pivot to be a random element of A
- Perform the partition

Until:
- resulting partition is such that $|L| \leq \frac{3}{4}|A|$ and $|R| \leq 3/4$

MARCH 2014

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

APRIL

Monthly Planner

Schedule your appointments, events and important dates

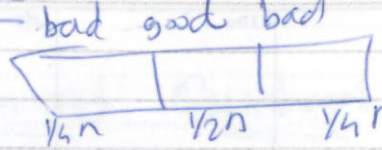
MAY 2014

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

1 Tue	2 Wed	3 Thu	4 Fri
5 Sat	6 Sun	7 Mon	8 Tue
9 Wed	10 Thu	11 Fri	12 Sat
13 Sun	14 Mon	15 Tue	16 Wed
17 Thu	18 Fri	19 Sat	20 Sun
21 Mon	22 Tue	23 Wed	24 Thu
25 Fri	26 Sat	27 Sun	28 Mon
29 Tue	30 Wed		

Action Plan for Previous Month

Analysis:

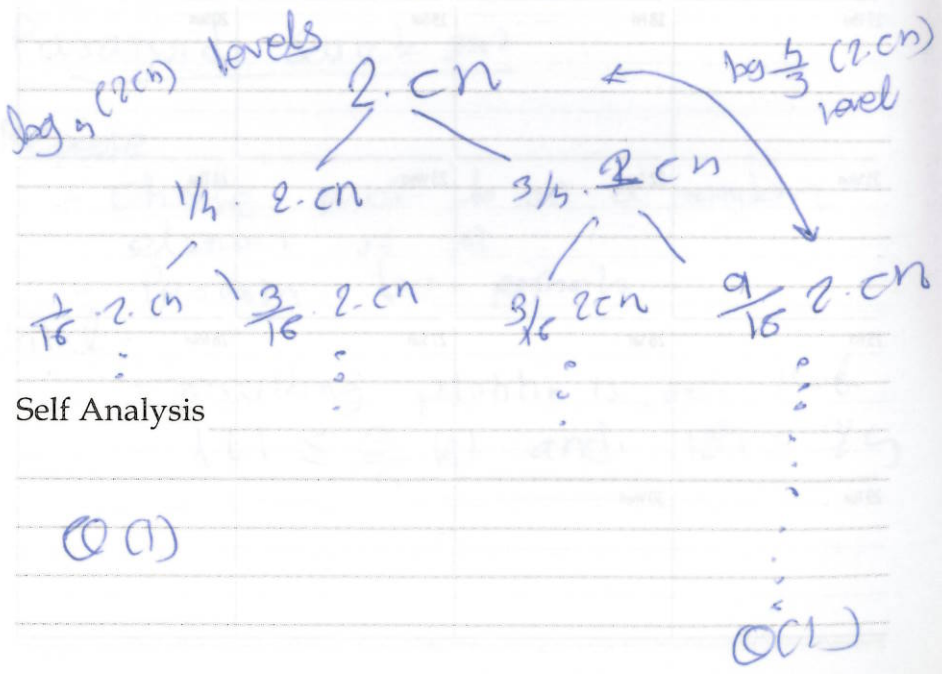


- probability of bad partition
- A call is good with prob $\geq 1/2$

$$T(n) = T(n/4) + T(3n/4) + \underbrace{\epsilon\left(\begin{array}{c}\text{# nodes in}\\ \text{set}\\ \text{good partn}\end{array}\right)}_{2} \cdot cn$$

$$= T(n/4) + T(3n/4) + 2 \cdot cn$$

Review



Self Analysis

ACTION PLAN FOR CURRENT MONTH

- unbalanced tree
  - $2cn$ work at each level
- ~~log~~ - max. $\log_{4/3} 2cn$ levels

$$T(n) = 2 \cdot cn \cdot \log_{4/3} 2cn$$

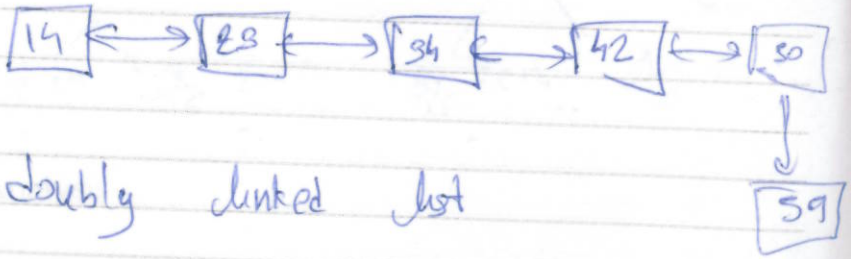
$$= n \log n$$

Lec-7

Skip-list

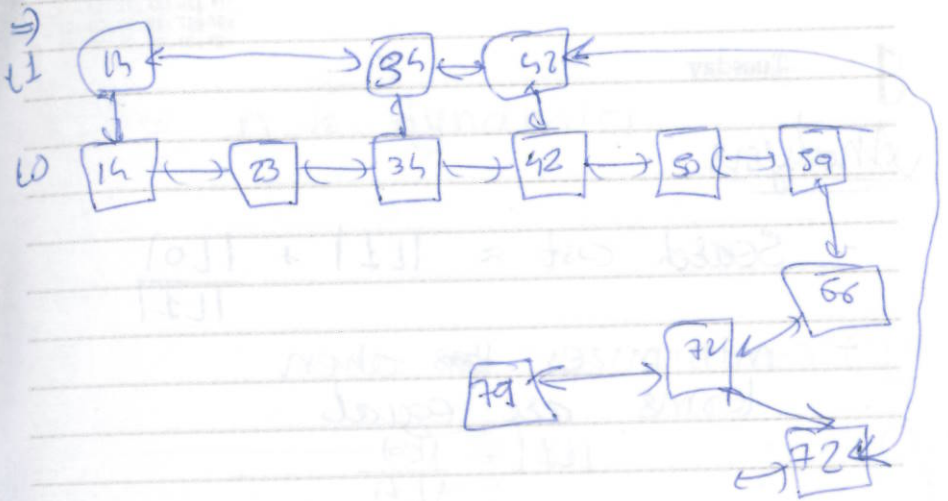
- Randomized data structure
- In addition to expectation, our analysis will also include "with high probability" (w.h.p)
- easy to implement than balanced tree
- It maintains a dynamic set of n elements in O(log n) time per operation in expectation and w.h.p

* one sorted link list search takes O(n) time



- doubly linked list

⇒ Let's have two lists



⇒ subway stops on 7th ave express line

* Search (66).
  - going from 34 to 42 L1
  - L1 to L0
  - 42 to 66 on L0

⇒ where to put express-stops ?

* Search (x) :. (2 linked list)
  - walk right on top linked list until next right would go too far
  - walk down to bottom list (L0)
  - Walk right on L0 until element is found (or not)

MARCH 2014

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

1 Tuesday

Analysis

- Search cost $\approx |L1| + \frac{|L0|}{|L1|}$
- minimizes when terms are equal

$$|L1| = \frac{|L0|}{|L1|}$$

$$\Rightarrow (L1)^2 = |L0| = n$$

$$\Rightarrow L1 = \sqrt{n}$$

2 Wednesday

- In our example n was 9 so $\sqrt{n} = 3$ elements in between

Complexity

2 sorted list $\Rightarrow 2\sqrt{n}$

3 sorted list $\Rightarrow 3\sqrt[3]{n}$

:

k sorted list $\Rightarrow k\sqrt[k]{n}$

log n sorted list $\Rightarrow (\log n)\sqrt[\log n]{n}$

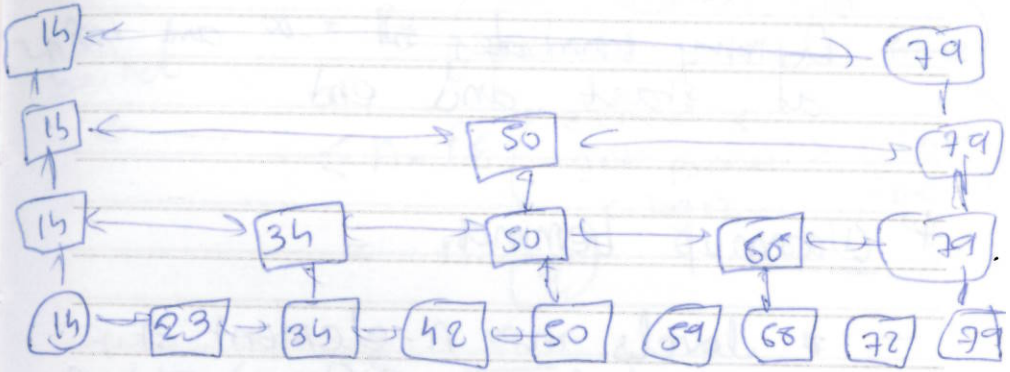
$= 2 \log n$

MAY 2014

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

3 Thursday

$\Rightarrow$ Now n is dynamic



4 Friday

Insert (x)

- To insert an element x into a skip list search (x) to see where x fits into bottom list
- Always insert into bottom list
- Insert into some of the lists above (which one?)

→ Flip fair coin:-

if heads:- promote x to next level up (may be newly created)
- repeat

else:
stop.

5 Saturday

Delete(x) :-
- search & delete at all levels.

→ Dummy nodes of -∞ and +∞ at start and end

Warmup Lemma

- # levels in n-element skip list is O(log n) w.h.p

6 Sunday

$c \cdot \log n$, prob $1 - \frac{1}{n^\alpha}$

→ $\alpha > 1$

- c is related to α
- as n increase prob. increases
- 80% prob of being in $2 \log n$
  99% prob of " " $4 \log n$

Monday 7

proof

- Failure probability
  = Pr(not $\leq c \cdot \log n$ levels)
  = Pr(> $c \cdot \log n$ levels)
  = Pr(some element got promoted > $c \cdot \log n$ times)
  ≤ n · Pr(element x got promoted > $c \cdot \log n$ time)
  = $n \cdot \left(\frac{1}{2}\right)^{c \cdot \log n}$
  = $\frac{n}{n^c}$
  = $\frac{1}{n^{c-1}}$
  = $\frac{1}{n^\alpha}$ $\alpha = c-1$

Union bound ←

out of n ↑

Tuesday 8

* Search Theorem : Any search in an n-element skip list costs O(log n) w.h.p.

-

9 Wednesday

* Analyze search backward

- b-search (backward search) starts [ends] at node in bottom list

At each node visited:
if node was not promoted (heads here) higher
then we go [come from] left

if node was promoted higher (heads here)
then we go [came from] up.

10 Thursday

Stop [start] when we reach top level ($+\infty$ $-\infty$)

* Proof theorem

- Backward search makes "up" moves and "left" moves each with prob = 1/2

Friday 11

- No. of moves going "up"
  $\leq$ # levels
  $\leq c \log n$ with h.p.

- Total no of moves
  = no. of moves till you get $c \log n$ up moves

claim → = no. of coin flips until $c \log n$ heads

= $O(\log n)$ wh.p

Saturday 12

* Chernoff theorem

Let y be a random variable representing total no of heads in a series of m independent coin flips, where each flip has $P(\text{head}) = p$. Then for all $\varepsilon > 0$, we have

$$Pr[Y \geq E[Y] + \varepsilon] \leq e^{-\frac{2\varepsilon^2}{m}}$$

13 Sunday

Lemma : for any c, there is a constant d s.t. w.h.p the no. of heads in flipping $d \log n$ fair coins is at least $c \log n$.

→ $d = 8c$ is enough. Last of notes

14 Monday

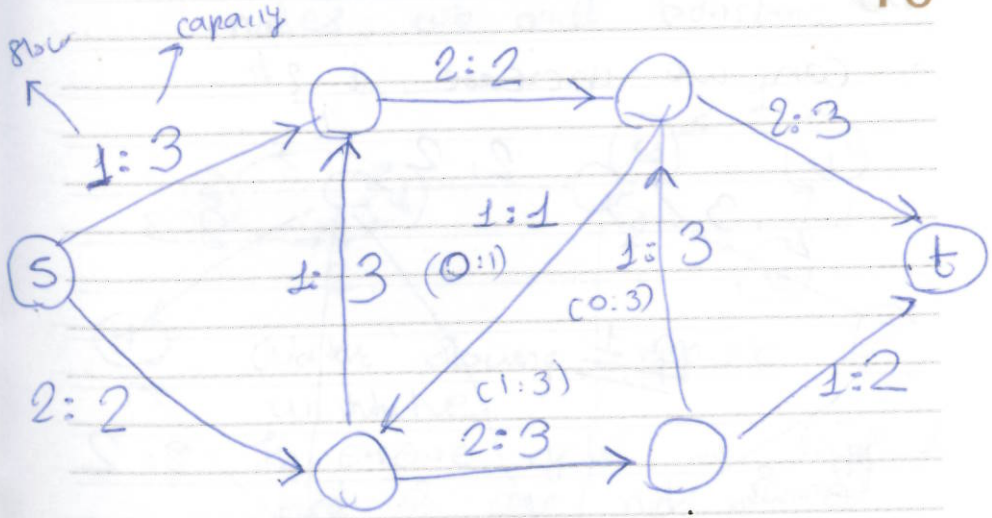
Tuesday 15

Lec-13 Incremental Improvement

⇒ Flow Networks

- $G(V, E)$
- directed graphs
- Two distinguish vertices source s, and sink t
- Each edge $(u, v) \in E$ has non negative capacity $c(u, v)$
- If $(u, v) \notin E$ then $c(u, v) = 0$

Wednesday 16



MARCH 2014

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

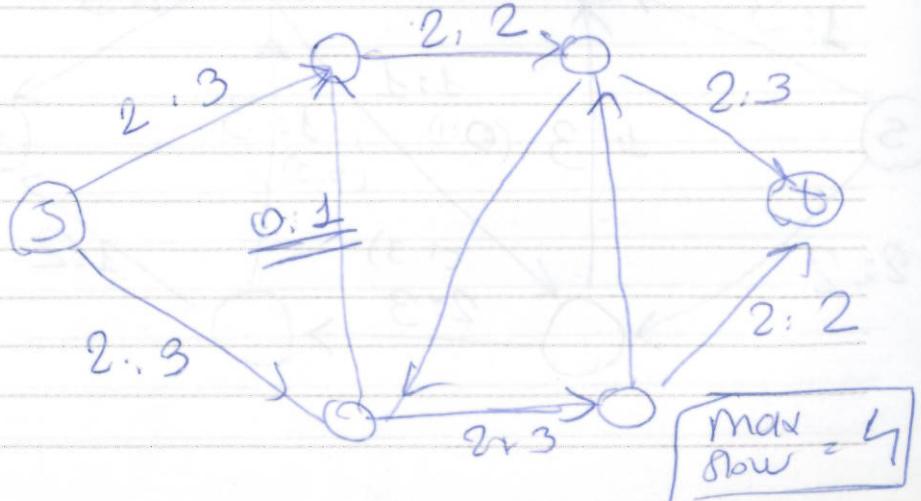
17 Thursday

- Cycles are around, but there will be other constraints we will talk later
- We can remove one unit from cycle and it still maintains flow
- In above example

$$\text{flow from source} = \text{flow from sink} = 3$$

18 Friday

- Can we increase it ?



19 Saturday

- We decrease a flow in some edges to increase overall flow.

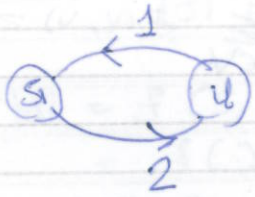
Problem

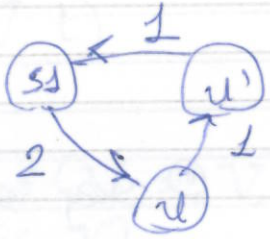
⇒ Given a flow network G, find a flow with maximum value on G.

⇒ Assumption
- No self loop is allowed

20 Sunday

- For a loop with two nodes, we will transform it





Water flowing opposite is un-intuitive.
- In reverse cycle of length 1 and 2 are not allowed.

21 Monday

- CLRS has two notions
  positive flow ≠ net flow
- Does not matter to us after above transformations

* **Flow (Net flow)**

Defination: A flow on G is function
$f : V \times V \to R$, satisfying :-

① capacity constraint :-
- for all $u, v \in V$,
  $f(u,v) \leq c(u,v)$

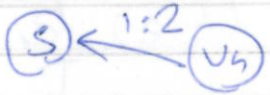
22 Tuesday

② flow constraint :-
- for all $u \in V - \{s,t\}$
  $\sum_{v \in V} f(u,v) = 0$

③ Skey symmetry :-
- for all $u, v \in V$,
  $f(u,v) = -f(v,u)$

- The value of a flow f, denoted $|f|$,
  $|f| = \sum_{v \in V} f(s,v) = f(s,V)$
  ↓ implicit summation notation

Wednesday 23



$f(s, v_4) = -f(v_4, s)$
$= -1$

* **Simple Properties**

→ $f(x,x) = 0$
$-f(a,b) + f(b,a) = 0$

→ $f(X,Y) = -f(Y,X)$

→ $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$
if $X \cap Y = \emptyset$

Thursday 24

Theorem $|f| = f(V,t)$

Proof:

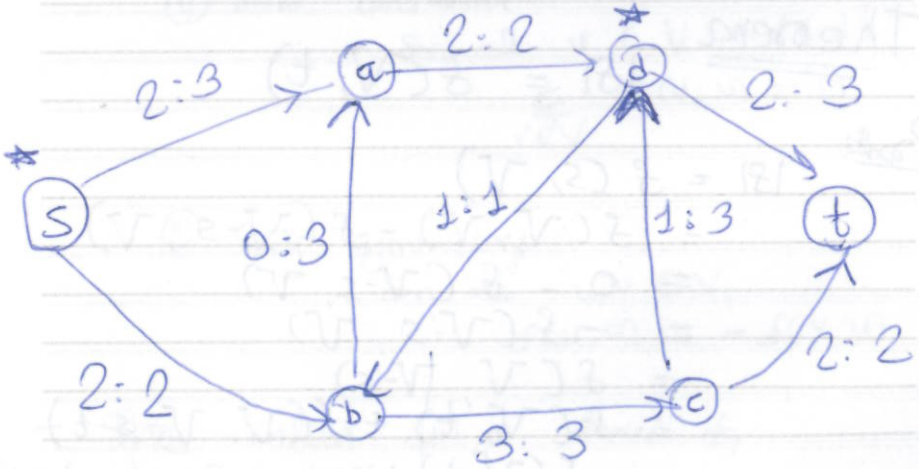
$|f| = f(s,V)$
$= f(V,V) - f(V-s,V)$
$= 0 - f(V-s,V)$
$= -f(V-s,V)$
$= f(V,V-s)$
$= f(V,t) + f(V,V-s-t)$
$= f(V,t) + 0$ (∵ flow constraint)
$= f(V,t)$

25 Friday

Cuts

- A cut (S, t) of a flow network $G \in (V, E)$ is a partition of V s.t. $s \in S$ and $t \in T$
- If f is a flow on G, then the flow across the cut is $f(S, T)$

26 Saturday



Sunday 27

$\star \in S$ (S, d)

$dest \in T$

$f(S, T) = $

$= (2 + 2) + (-2 + 1 - 1 + 2)$

$sa \quad sb \quad da \quad db \quad dc \quad dt$

capacity of cut $= c(S, T)$

$= (3 + 2) + (1 + 3)$

$sa \quad sb \quad db \quad dt$

$= 9$

Monday 28

⇒ Value of any flow is bounded by capacity of any cut

⇒ Another characterization of flow value:-

Lemma:- For any flow f and any cut (S, T) we have $|f| = f(S, T)$

Proof:-

$f(S, T) = f(S, V) - f(S, S)$

$(\because S \cup T = V)$

APRIL 2014

MARCH 2014

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

29 Tuesday

$= f(S, V) + 0$

$= f(S', V)$

$= f(S, V) + f(S' - S, V)$

$= f(S, V)$ ↓ does not contain t

$= |f|$ (and we have taken s out)

- flow through cut = flow thrust return
  ⇒ capacity of any cut can bound the flow of network

30 Wednesday

- we are going towards min-cut-max-flow theorem/algorithm

* Residual Network — original network $G(V, E)$

- $G_f(V, E_f)$
  - strictly positive residual capacities

$c_f(u, v) = c(u, v) - f(u, v) > 0$

edges in E_f admits more flow

APRIL 2014

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

MAY

Monthly Planner

Schedule your appointments, events and important dates

JUNE 2014

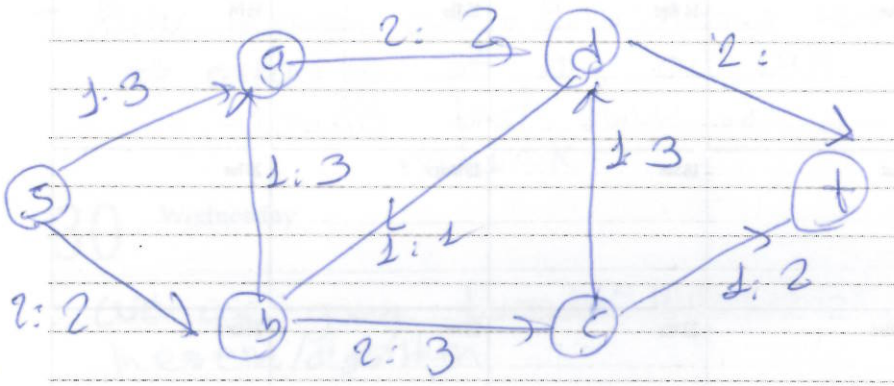
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

1 Thu	2 Fri	3 Sat	4 Sun
5 Mon	6 Tue	7 Wed	8 Thu
9 Fri	10 Sat	11 Sun	12 Mon
13 Tue	14 Wed	15 Thu	16 Fri
17 Sat	18 Sun	19 Mon	20 Tue
21 Wed	22 Thu	23 Fri	24 Sat
25 Sun	26 Mon	27 Tue	28 Wed
29 Thu	30 Fri	31 Sat	

Action Plan for Previous Month

- If $(v, u) \notin E$, $c(v, u) = 0$,
  but $f(v, u) = -f(u, v)$

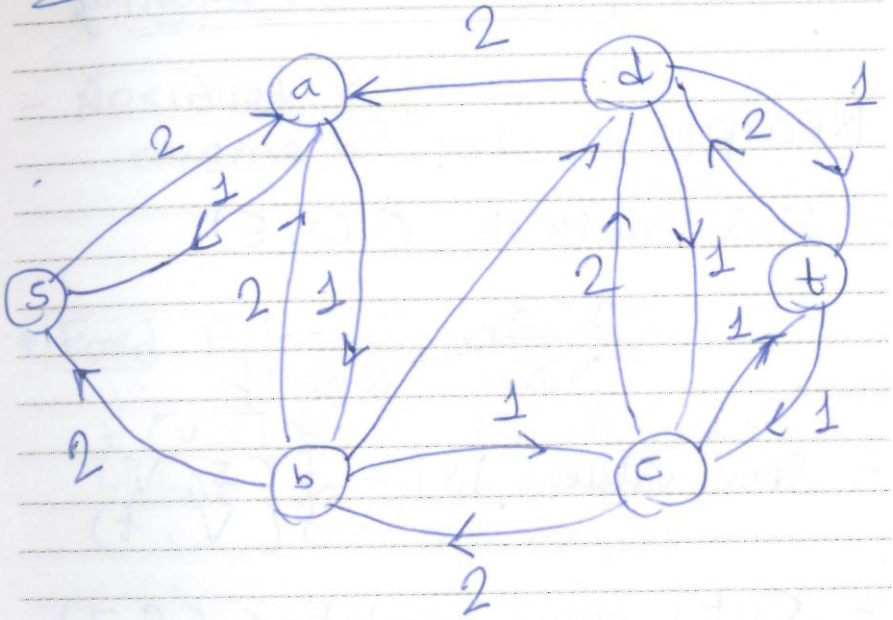
Review



Self Analysis

ACTION PLAN FOR CURRENT MONTH

G_f



- Augmenting path in G_f
  - Path from s to t in G_f
  - If such a path exists flow can be increased
    - path can be found by DFS, BFS
  - $s \to a \to b \to c \to t$
    $\min(2 \quad 1 \quad 1 \quad 1)$
    $= 1$
  - Add/subtract 1 in orginal path by doing residue network again

Lec-14 Incremental Improvement : Matching

* Recap:-

- Flow network $G(V, E)$

$1:3$ → (1 = flow, 3 = capacity)

- Flow value: $|f| = f(s, V) = f(V, t)$

- Cut: any partition (S, T) s.t. $s \in S$, $t \in T$

- Lemma: $|f| = f(S, T)$ for any cut (S, T)

- Corollary: $|f| \leq c(S, T)$ for any cut (S, T)

- Residual graph: $G_f(V, E_f)$ with strictly positive residual capacities $c_f(u, v) = c(u, v) - f(u, v) > 0$

- Augmenting path :- Any path from s to t in G_f

- Residual capacity :- of an augmented path $c_f(P) = \min_{(u,v) \in P} c_f(u, v)$

* Ford Fulkonson Algorithm

$f[u, v] \leftarrow 0$ for all (u, v)
while an augmenting path in G_f exists:
    do augment f by $c_f(P)$

G :: $s \xrightarrow{1:2} a \xrightarrow{1:4} t$

G_f :: $s \xrightarrow{1} a \xrightarrow{3} t$, $a \xrightarrow{1} s$, $t \xrightarrow{1} a$

$c_f(P) = 1$

G :: $s \xrightarrow{2:2} a \xrightarrow{2:4} t$

G_{f_1} :: $s \xrightarrow{2} a \xrightarrow{2} t$, $a \xrightarrow{2} s$, $t \xrightarrow{2} a$

No augmenting path in G_{f_1}

1 Thursday

Max Flow Min Cut Theorem

Theorem:- The following are equivalent-

① $|f| = c(S,T)$ for some cut (S,T)
② f is a maximum flow
③ f admits no augmenting paths

→ we need 3 ⇒ 2
→ what we will do is
1 ⇒ 2
2 ⇒ 3
3 ⇒ 1

2 Friday

Proof:-

1 ⇒ 2 since $|f| \leq c(S,T)$ for any cut (S,T), the assumption that $|f| = c(S,T)$ implies f is a max. flow (can not be increased)
→ edge capacity

Saturday 3

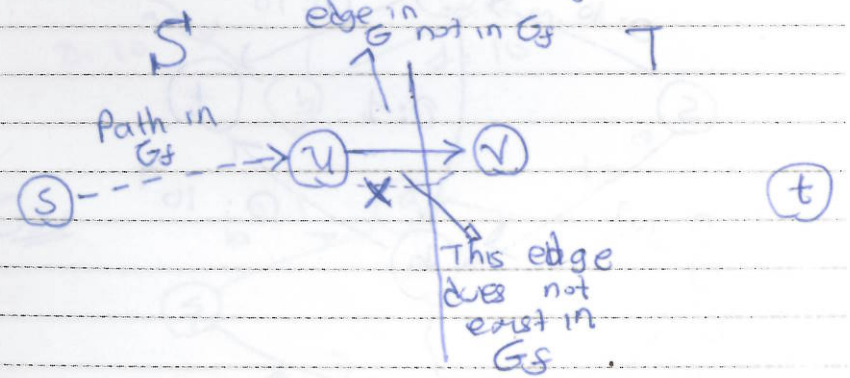
2 ⇒ 3
- If there were an augmented path, the flow value could be increased, contradicting maximality of f.

3 ⇒ 1
- Suppose f admits no augmenting path (lack of connectivity between s and t)

Define $S = \{u \in V$: there exists a path in G_f from s to $u\}$
$= \{$ reachable vertices from s $\}$

Sunday 4

$T = V - S$ [$t \in T$, $s \in S$ ⇒ (S,T) is cut]



MAY 2014

APRIL 2014

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

5 Monday

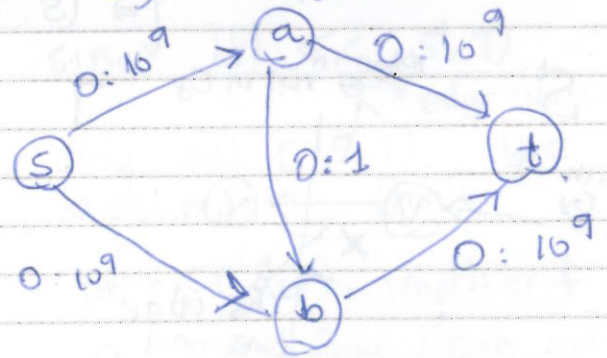
→ $c_f(u,v) = 0$, since
if $c_f(u,v) > 0$ then
$v \in S'$ not $v \in T$
as assumed

- Thus $f(u,v) = c(u,v)$
because
$c_f(u,v) = c(u,v) - f(u,v) = 0$

- Summing over all $u \in S'$ and $v \in T$ yields
$f(S',T) = c(S',T)$

6 Tuesday → Above is why Ford-Fulkonson works.

* Complexity Analysis is remaing



JUNE 2014

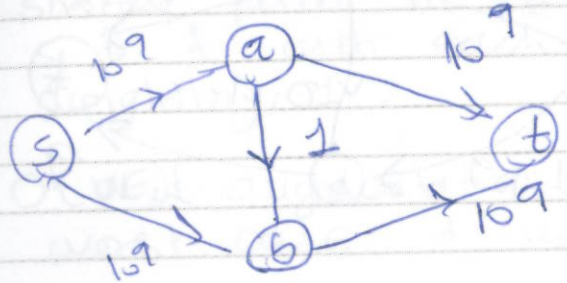
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

2014 MAY

Wednesday 7

- Ford Fulkonson fails for above ~~path~~ graph

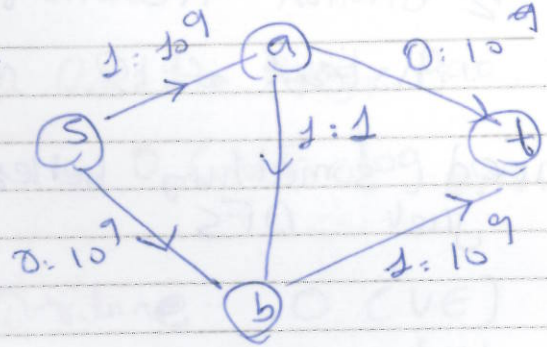
Cf



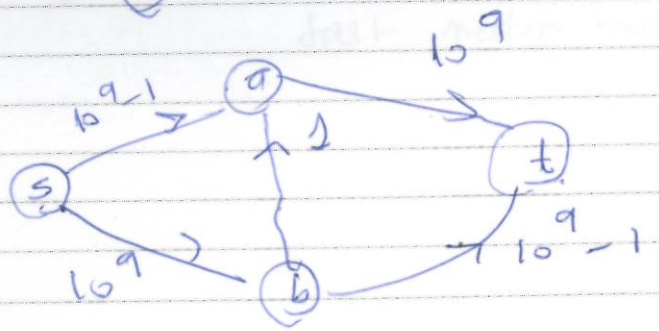
⇒ we picked $s \to a \to b \to t$

Thursday 8

G:



Gf1

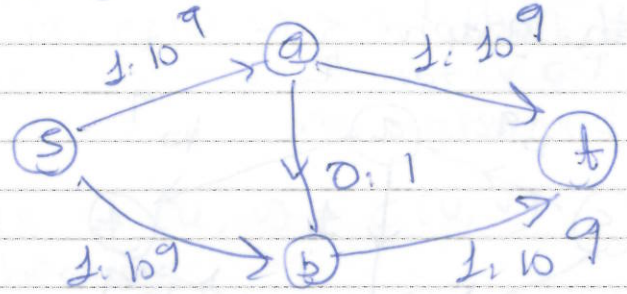


APRIL 2014

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

9 Friday

G: → B.



→ With wrong choice of path we end up with 2 billion iteration instead of 2

10 Saturday

- We need something better than just DFS

JUNE 2014

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

Sunday 11

Edmonds Karp

→ BFS augmenting path is a shortest path in G_f from s to t with each edge weighting 1.

→ $O(VE)$ augmentation in the worst case if used above BFS

overall complexity : $O(VE^2)$

Monday 12

* In 2011 :- King, Rao, Tarjan

$O(VE \log_{E/V \log V} V)$

* Orlin :- $O(VE)$ fast matrix multiply

APRIL 2014

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

13 Tuesday

* Baseball Elimination

- Algorithm to look at standing, and see if there is still a chance to make to play-offs.

14 Wednesday

Team	wins w_i	losses l_i	to play g_i	against each other g_{ij}				
1. NY	75	59	28	-	5	7	4	3
2. Baltimore	71	63	28	5	-	2	4	4
3 Boston	69	65	28	7	2	-	4	0
4. Toronto	63	71	28	4	4	4	-	0
5. Detroit	0	0	28	3	4	0	0	-

NY Balti Boston Toron Detroit

JUNE 2014

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

Thursday 15

* Team i is eliminated if $w_i + g_i < w_j$ for some j

⇒ Detroit:-
$w_5 = 46$, $l_5 = 88$

$46 + 28 = 74 < 75$
hence detroid is eliminated

- Above is sufficient but not necessary

* What if $w_5 = 47$

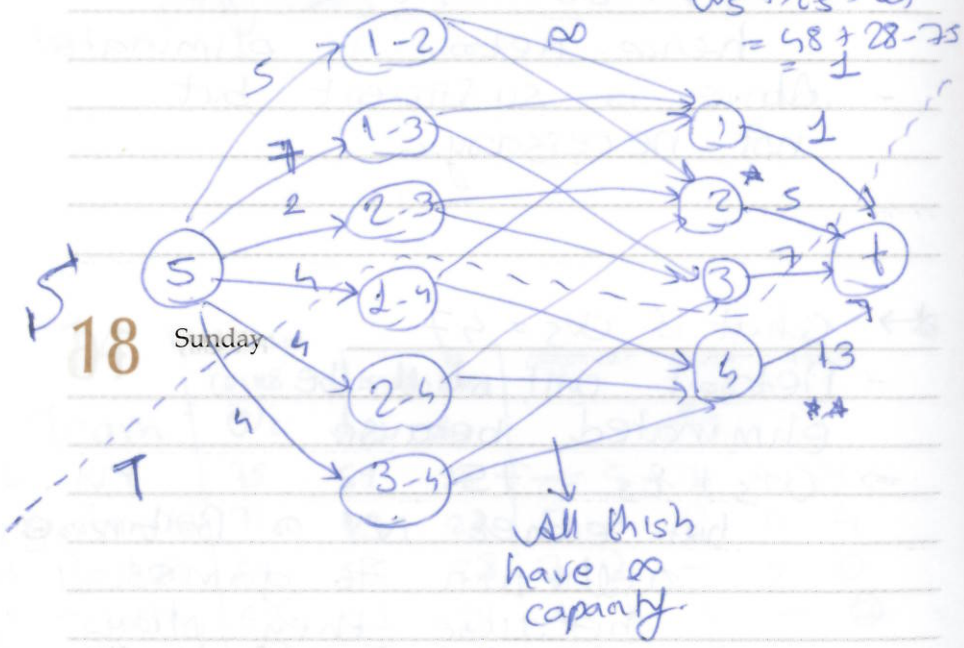
Friday 16

- Detroit will still be eliminated because
⇒ $w_5 + g_5 = 75$
but either NY or Baltimore will win 76 games because they play each other 5 times

17 Saturday

* What if $\omega_5 = 48$

→ Below is a flow network to determine if team 5 is eliminated.



18 Sunday

* $\omega_5 + \ell_5 - \omega_2 = 5$

** ~~No. are capacity~~

Capacities are # games team i can win and not have more wins than team 5

Monday 19

Intuition:- Assume team 5 wins all remaining ~~edges~~ games. Deviate up remaing games so all teams have $\leq \omega_5 + \ell_5$ wins

- If you can do above team 5 is eliminated else it is not.

Theorem:- Team 5 is eliminated if and only if max flow does not saturate all edges leaving the source. (i.e min cut < 26)

Tuesday 20

(Saturation corresponds to playing all remaining games)

Argument: If you can't play all remaining games without exceeding capacity of $i \to t$ edges team 5 is eliminated.

21 Wednesday

* Find min-cut (dotted line)

5 → 7 edges

4 + 4 + 4 + 1 + 5 + 7 = 25

⇒ which implies elimination.

22 Thursday

Friday 23

Q-1 how do we avoid stucking at local minima / saddle point in NN ?

* I have seen ge

Saturday 24

APRIL 2014

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

25 Sunday

Linear Programming

⇒ variables $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$

⇒ objective fn : $\vec{c} \cdot \vec{x} = c_1 x_1 + \dots + c_n x_n$

⇒ inequalities : $A\vec{x} \leq \vec{b}$

→ Max. $\vec{c} \cdot \vec{x}$
s.t. $A\vec{x} \leq \vec{b}$
and $\vec{x} \geq 0$

26 Monday

Certificate of Optimality

→ Duality helps proving optimality

→ Is there a short certificate that shows LP soln is optimal ?

consider $\frac{25}{222}$ (1) + $\frac{46}{222}$ (2) + $\frac{1}{222}$ (3) ← eg constr

$\Rightarrow x_1 + x_2 + \frac{140}{222} x_3 + x_4 \geq \frac{3100.000}{111}$

$(x_1 + x_2 + x_3 + x_4 \geq x_1 + x_2 + \frac{140}{222} x_3 + x_4)$

JUNE 2014

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

Tuesday 27

$\Rightarrow x_1 + x_2 + x_3 + x_4 \geq \frac{3100\,000}{111}$

→ hence it is the minimum amount to spend

LP Duality

Theorem :- $\max \vec{c} \cdot \vec{x}$
s.t. $A\vec{x} \leq \vec{b}$, $\vec{x} \geq 0$
primal →
111

Dual

Wednesday 28

$\min \vec{b} \cdot \vec{y}$
s.t $A^T \vec{y} \geq \vec{c}$
$\vec{y} \geq 0$

Converting to standard form

(1) minimize $-2x_1 + 3x_2$
$\max 2x_1 - 3x_2$

(2) x_j does not have non-negativity constraint
x_j replace with $x_j' - x_j''$
$x_j' \geq 0$
$x_j'' \geq 0$

29 Thursday

3) equality constraint $x_1 + x_2 = 7$

① $x_1 + x_2 \leq 7$

$x_1 + x_2 \geq 7$

③ $-x_1 - x_2 \leq -7$

⑤ $\geq$ constraint multiply by (-1)

→ Reduction:
people spend week of time reducing thres problem to LP

30 Friday

Maximum Flow

$$\max \sum_{v \in V} f(s,v) = |f|$$

st.

skew symmetry $f(u,v) = -f(v,u) \quad \forall u,v \in V$

conservation $\sum_{v \in V} f(u,v) = 0 \quad \forall u \in V - \{s,t\}$

capacity $f(u,v) \leq c(u,v) \quad \forall u,v \in V$

Saturday 31

→ We don't get performance improvement with LP, as you have specialized solution for max flow

→ These comes multi-comodity max flow problem
- Two comodities – 1 & 2
- f_1, c_1 and f_2, c_2
  - two distinct disjoint opt n
- f_1, f_2 and single capacity C
  $f_1(u,v) + f_2(u,v) \leq c(u,v)$
- Above does not have specialized solution

Shortest Path

- Single source shortest path from vertex s

$$\max \sum_{v \in V} d[v]$$

st. $d[v] - d[u] \leq w(u,v) \quad \forall u,v \in E$

$d[s] = 0$

→ we already have a min in inequality.