Distributed Algorithms

- Parallel Computing (multicore processors)
  - whether and how a task can be parallelized
- Distributed computing
  - networks/internet

n processors, w/ own input x_i
we want $y_i = f(x_1, x_2 \ldots x_n)$

1) Message passing model
   (each vertex has edges w/ port name)
2) Shared memory model

Leader Election

Goal: Run a protocol ST at termination, exactly one processor announces "I am leader"

- If we have 2 processors that are identical, then vibes

Solution I: Make processors non-identical (like IP, MAC, etc)

Simple protocol: Each processor has $max_i = MAX[max_i, \{\text{incoming msgs}\}]$
(with are everyone else's max_i)

After Δ rounds, if $max_i = ID_i$, output "I am leader"

Solution II: No unique IDs, but have randomness

Then alive protocol:

If all IDs are unique we good

Prob of collision: $Pr[ID_i = ID_j] = 1/k$

By union bound $Pr[\text{collision}] \leq \sum Pr[ID_i = ID_j] = \binom{n}{2}\frac{1}{k} \leq \varepsilon$

So if $k \geq \varepsilon^{-1}\binom{n}{2}$, succeeds w/ prob $\geq 1-\varepsilon$

Improve to LV alg → before announcing leader, check to see if only one, or repeat

→ expect $\leq \frac{1}{1-\varepsilon}$ repeats

→ # of rounds $\Theta\left(\frac{\Delta}{1-\varepsilon}\right)$, in Δ rounds

Maximal Independent Set

Goal: Protocol ends w/ each processor reaching a yes/no decision ST yes processors are maximally ind.

→ NO 2 yes processors are neighbors

→ maximal, can't add any more

Leader election

→ Set leader to yes, neighbors to NO

→ Repeat for undecided vertices $\Rightarrow O(n\Delta)$

Luby's MIS Protocol

- All processes active at start
- Phases
  - Round 1:
    Choose random value $r_i \in \{1, 2, \ldots k\}$ & send to all neighbors
    If all received values are all $< r_i$, join MIS (set output to yes)
  - Round 2:
    - If you joined MIS, announce to all neighbors
    - If you receive such an announcement, decide not to join MIS
    - If you set output (Yes/No) in this phase, become inactive

- Final set is independent
- Final set is maximal — the only way to become inactive is if you or one of your neighbors joins MIS

How many rounds until done?

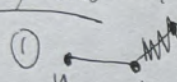
$\rightarrow \Theta(\log n)$! , Pairs k (random range) $>> n^3$

Pf:
if $i \neq j$, then $Pr[r_i = r_j] = \frac{1}{k} << \frac{1}{n^3}$

$Pr[\text{Collision}] \leq \frac{4 \log n \binom{n}{2}}{k} << \frac{1}{n}$ if $k \geq n^4$

(# pairs)

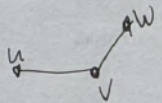
Pf by Cases: Say an edge (u,v) is still active iff both u, v still active

(1) 

$r_u > r_v$ or vice versa ½ prob

(u,v) will become inactive

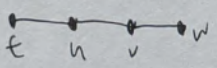
$Pr[u,v \text{ inactive}] = 1$

(2) 

one active edge incident to (u,v)

$Pr[u,v \text{ inactive}] \geq 1/2$

(3) 2 active edges incident to (u,v)



$Pr[r_u < r_v \wedge r_w < r_v] = \frac{1}{4}$

$Pr[r_u > r_v \wedge r_t < r_u] = \frac{1}{4}$

$Pr[u \text{ or } v \text{ is local max}] = 1/2$

(u,v) becomes inactive w/ prob $\geq 1/2$

So all $\forall Pr[u,v \text{ is still active after } \ell \text{ phases}] \leq \left(\frac{1}{2}\right)^\ell$

→ Union Bound $|E|$ edges

$Pr[\text{protocol terminated after } \ell \text{ phases}] = Pr[\text{some } (u,v) \text{ still active after } \ell \text{ phases}] \leq |E| \left(\frac{1}{2}\right)^\ell$

$\leq \frac{n}{2^\ell} \leq \frac{1}{n^3} << \frac{1}{n}$

if $\ell \geq 4 \log n$

Approx Algs – solve hard problems in quickly. Allow suboptimal solutions.

- Optimization pb of size n: C^* = cost of optimal solution
  C = cost of approx solution

ratio bound $\rho(n)$: $\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \le \rho(n)$, $\forall n$ $\Rightarrow$ $\rho(n)$-approx alg

(min this, max this)

- Approx scheme: input $\varepsilon > 0$, provides $(1+\varepsilon)$-approx alg
  PTAS: provides poly(n) alg, not necessarily in $\frac{1}{\varepsilon}$ $\left(O(n^{2/\varepsilon})\right)$
  FPTAS: alg poly in n and $1/\varepsilon$ $O\left(\frac{n}{\varepsilon^2}\right)$

Ex Vertex Cover (we saw decision version)

Input: $G = (V, E)$
Output: Set of vertices $S \subseteq V$ ST $\forall e = (u,v) \in E$, $S \cap e \neq \emptyset$
Obj: minimize $|S|$

2-approx for VC:

Alg: Pick any edge $(u,v) \in E$. Add both u & v. Remove all edges from E incident on u, v. Terminate when $E = \emptyset$

→ Running time: $O(V+E)$
→ Nondeterministic
→ But also always valid ☺

Notes
- Picking just one endpt is not an improvement
- Greedier in vertex degree does not necessarily help.
- Showed 2-approx w/o determining $|S_{OPT}|$

CLAIM: $|S_{APX}| \le 2|S_{OPT}|$

Pf: A be set of edges selected by graph
- $|S_{APX}| = 2|A|$
- optimal vertex cover must include at least one endpoint for each edge in A
  $|A| \le |S_{OPT}|$
- $|S_{APX}| \le 2|S_{OPT}|$

Ex Set Cover (NP-hard)

Input: X (set) of n pts, m subsets S_i of X ST $\bigcup_{i=1}^{m} S_i = S_1 \cup S_2 \cdots = X$
Find: cov $C \subseteq \{1, \dots, m\}$ ST $\bigcup_{i \in C} S_i = X$ while $|C|$ is minimized.
(Smallest # of subsets to cover all vertices)

Greedy: Repeat until all elem covered
- Choose S_i to maximize # uncovered elems
- Add i to C
- Mark all elem from S_i as covered

Runtime: $\Theta(\min(n,m))$, each it is $\Theta(mn)$
Product: $\Theta(mn \cdot \min(m,n))$

Claim: $(\ln(n)+1)$-approx
→ Let $t = |C_{OPT}|$
→ Let X_i be set of remaining elems. X_i can be covered by t sets or fewer

∃ a set covers $\ge \frac{|X_i|}{t}$ elem ← pick

So $|X_{i+1}| \le \left(1 - \frac{1}{t}\right)|X_i|$

$|X_i| \le \left(1 - \frac{1}{t}\right)^i |X| = \left(1 - \frac{1}{t}\right)^{t \ln n} \cdot n \le e^{-\ln n} n = 1$

(for $i = t \ln n$ → we terminate)

Partition

Input: Sorted list of n positive #s $S_1 \geq S_2 \geq \dots S_n$

Output: A partition of the indices $\{1 \dots n\}$ into two sets A and B ST

① $\{1, \dots n\} = A \cup B$

② $\max\left\{\sum_{i \in A} S_i, \sum_{i \in B} S_i\right\}$ is minimized

APPROX ALG

Define $m = \left\lceil \frac{1}{\varepsilon} \right\rceil - 1$, $\varepsilon \geq \frac{1}{m+1}$

$O(2^m)$ — First Phase: Find optimal partition A', B' for the first m numbers

$O(n)$ — Second Phase:

```
Set A ← A' & B ← B'
for i = m+1 to n
    if w(A) ≤ w(B): A ← A ∪ {i}
    else: B ← B ∪ {i}
return (A, B)
```

How close to optimal?

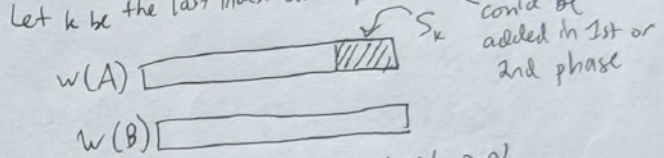
- This is a $(1+\varepsilon)$ approx alg

WLOG: Let $w(A) \geq w(B)$

$2L = \sum_{i=1}^{n} S_i$, $w(A) \geq L$, $w_{OPT}(A) \geq L$

Define approx ratio as $\frac{w(A)}{L}$

Let k be the last index added to set A



Case 1: k added in 1st phase → so $A = A'$, $B = B'$, we have OPT part.

Case 2: second phase:

$w(A) - S_k \leq w(B) = 2L - w(A)$

$w(A) \leq L + \frac{S_k}{2}$ $\quad S_1, S_2 \dots S_m \geq S_k$

$2L \geq (m+1) S_k$ $\quad k > m$

Approx ratio $= \frac{w(A)}{L} \leq 1 + \frac{S_k}{2L} \leq 1 + \frac{1}{m+1} \leq 1 + \varepsilon$

Hashing

u = # keys in universe

n = # keys in table

m = # slots

$h: \{0,1,\dots u-1\} \to \{0,1,\dots m-1\}$

Simple uniform hashing: $\Pr_{k \neq k'}\{h(k)=h(k')\} = \frac{1}{m}$ $\quad \Theta(1+\alpha)$ /op, $\alpha = \frac{n}{m}$

what if keys aren't random? choose hash functions randomly.

UNIVERSAL HASHING

$\mathcal{H}$ is a uni hash fam if $\Pr_{h \in \mathcal{H}}\{h(k)=h(k')\} \leq \frac{1}{m}$ for all k, k' — To prove something isn't uni, take a $h(k,k')$ and prove w/ random h's, it collides a lot.

$\mathbb{E}[\#\text{ keys in a slot}] \leq 1+\alpha$, $\alpha = \frac{n}{m}$

Dot-product hash family: $h_a(k) = a \cdot k \bmod m$ $\quad a = \langle a_0, a_1, \dots a_{r-1}\rangle$

Pf universal: $\Pr_{a \in \mathcal{H}}[h_a(k)=h_a(k')] = \Pr[\sum a_i k_i = \sum a_i k_i' \text{ (mod } m)]$

$= \Pr\left[\sum_{i \neq d} a_i k_i + a_d k_d \right] = \sum a_i k_i' + a_d k_d' \bmod m$ (where k, k' differ)

$= \Pr\left(\sum a_i(k_i - k_i') + a_d(k_d - k_d') \overset{?}{=} 0 \bmod m\right)$

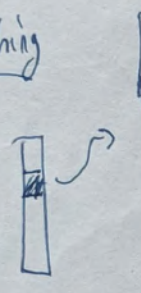
$= \Pr\left[a_d = -(k_d - k_d')^{-1} \sum_{i \neq d} a_i(k_i - k_i') \bmod m\right]$

$= \mathbb{E}_{a_{\text{not } d}}\left[\Pr_{a_d}\{a_d = f(k,k', a_{\text{not } d})\}\right]$

$= \sum_x \Pr\{a_{\text{not } d} = x\} \Pr_{a_d}\{a_d = f(k,k',x)\}$

$= \mathbb{E}\left[\frac{1}{m}\right] = 1/m$

Perfect Hashing



1 regular hash table $m = \Theta(n)$

another hash table, but ℓ_j^2 slots, ℓ_j = # elems in slot j of 1st

- if $\sum_{j=0}^{m-1} \ell_j^2 > cn$, hash again
- if for some $h_{2,j} \neq h_{2,j}(k')$, recreate $h_{2,j}$

$I_{i,i'} = \begin{cases} 1 & \text{if } h(k_i) = h(k_{i'}) \\ 0 & \text{ow} \end{cases}$

$\mathbb{E}\left[\sum \ell_j^2\right] = \mathbb{E}\left[\sum_{i=1}^{n}\sum_{i'=1}^{n} I_{i,i'}\right]$

$= \sum \sum \mathbb{E}[I_{i,i'}]$

$= n + 2\binom{n}{2} \cdot \frac{1}{m}$

($k=k'$; $k \neq k'$; Pr 2 hash together)

$= O(n)$, $m = \Theta(n)$

$\Pr[\text{collision}] \leq 1/2$ by Markov

Random walks – we care about distribution after

p_v^t = prob of being at v at time t

$p_v^{t+1} = \sum_{e=(u,v)} \frac{1}{d(u)} p_u^t$

$A_{uv} = \begin{bmatrix} 1 \text{ if } (v,u) \in E \\ 0 \text{ ow} \end{bmatrix}$ $D = \begin{bmatrix} d(u) \text{ if } u=v \\ 0 \text{ ow} \end{bmatrix}$ (D^{-1} is $1/d(u)$)

if weighted, use w_e

$W = AD^{-1} \Rightarrow \begin{cases} 1/d(v) \text{ if } (v,u) \in E \\ 0 \text{ ow} \end{cases}$

Lazy random walks – can remain at current vertex. (self loops)

$p_v^{t+1} = p_{lazy} p_v^t + (1 - p_{lazy}) \sum_{e=(u,v)} \frac{1}{d(u)} p_u^t$

$p^{t+1} = W p^t = W^{t+1} p_0$; Non lazy

$\hat{W} = p_{lazy} I + (1 - p_{lazy}) W$; lazy

$\hat{p}^{t+1} = \hat{W} \hat{p}^t = \hat{W}^{t+1} \hat{p}_0$

Does this always converge to stationary dist?

Check

what is stationary dist? Write out $\pi_0 \to \pi_1$, $\pi_1 \to \pi_2$, $\pi_2 \to \pi_3$ .. Solve for π_3 in terms of π_0 ...

Stationary = π

$W\pi = \pi$

$\pi_v = \frac{d(v)}{\sum_{u \in V} d(u)}$

Every connected, non bipartite undirected graph has a stationary distribution to which random walks converge.

– Lazy: can even be bipartite

– Directed: Strongly connected, aperiodic

↓

If you can prove ∃ self loops, you're good

Markov chain – process that depends on current state

– Metropolis-Hasting: construct transition probabilities on the fly

– random walks → stationary distn.

Pairwise ind. $\mathcal{H}$

$\Pr_{h \in H, x_1 \neq x_2} [h(x_1) = y_1 \wedge h(x_2) = y_2] = \frac{1}{|Y|^2}$

Streaming – limited space

Reservoir sampling – For each i, we want to output x_i w/ prob $1/n$, but we don't know n!

– keep each elem we already have w/ prob $\frac{i}{i+1}$, new elem $\frac{1}{i+1}$

Sample k random: Include first k, at each x_{i+1}, keep w/p $\frac{k}{i+1}$, remove random elem from the k in reservoir

Frequency moments

F_0 = # distinct elems

n elem total

d distinct elems

```
z = 0
for each j in list:
    if zeroes(h(j)) > z
        z = zeroes(h(j))
```

return 2^z ↑ F_0

– No add space required

Graph spanner: Reduce size of graph while approx preserve distances.

$G = (V, E)$ – subgraph $H = (V, E_H)$, α spanner of G if

$E_H \subseteq E$

$d_G(u,v) \leq d_H(u,v) \leq \alpha \cdot d_G(u,v)$ $\forall u,v$

rec 8 alg

– If G_1 and G_2 are 2 edge-disjoint graphs over a set of vertices $V, H_1, \& H_2$ are $(1+\varepsilon)$-sparsifiers of G_1, G_2 respectively, then $H_1 \cup H_2$ is $(1+\varepsilon)$-spars. of $G_1 \cup G_2$

– If $G_1 \to \alpha$ spars of G, $G_2 \to \beta$ spars of G_1 $\to G_2$ is $\alpha\beta$ spars of G

– Build H iteratively, then Div + Conquer (Rec 8, fig 2)

Randomized Alg

- Monte Carlo – runs fast, probably correct
- Las Vegas – probably runs fast, correct

Ex Matrix Product

$A \cdot B = C$?

$A(B \cdot r) = C \cdot r$?

n^2 ↓ n^2

If $A \cdot B$ does in fact $= C$, then it will produce $=$.

If not, $D = C - AB$, some D is nonzero.

$\exists\, Dr \neq 0$, for some r

Say $D[i][j]$ is nonzero

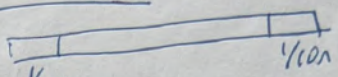
we need $r[j]$ to be nonzero.

$r_{bad} + V = r_{good}$

where $v[j]$ is 1, ow 0

$\exists$ good and bad r for every $j \rightarrow$ 1/2 of r's are good.

Randomized Select – find elem of rank k



- Select random x
- Partition around it $O(n)$
- if $i = k$: return
- else: recurse on proper side

Analysis: if we reduce 9/10 every time

$$T = \sum_{r=0}^{\log n} \Theta\left(\left(\frac{9}{10}\right)^r n\right)$$

But not all the time in reality...

$$T = \sum \Theta\left(\left(\frac{9}{10}\right)^r n\right) \quad T_r,\ T_r > s \quad Pr \leq 0.2^s$$

$E(T_r) \leq \frac{5}{16}$ ↓ # times to get to an element in range

so $E(T) = O(n)$

Expectation → Chernoff Bounds for bounding probs

Linearity of Expectation: $E[X] = E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]$

Union Bound – $Pr\left[\bigcup_{i=1}^{n} A_i\right] \leq \sum_{i=1}^{n} Pr(A_i)$ (selecting either of elems vs selecting just one)

Markovs Ineq → $Pr[Y \geq a] \leq \frac{E[Y]}{a}$ – above a threshold

Chebyshevs Inequality: $Pr[|X - \mu| \geq k] \leq \frac{\sigma^2}{k^2}$ (var) – outside some k from μ

expect. some value

Chernoff Bound

$Pr[X > (1+\beta)\mu] < e^{-\beta^2/3}$ $0 < \beta < 1$

$Pr[X > (1+\beta)\mu] < e^{-\beta^2\mu/3}$ $\beta > 1$

$Pr[X < (1-\beta)\mu] < e^{-\beta\mu/2}$ $0 < \beta < 1$

Ex: biased coin that is heads 1/3 total n times

Determine a value of n s.t. probability of getting more than half of the flips heads is $< 1/1000$.

- $X_i = 1$ if ith coin is heads, 0 ow.
- $X = \sum X_i$. $Pr[X > \frac{n}{2}] < \frac{1}{1000}$, find n
- $E(X) = n/3$

$\beta = \frac{1}{2}$

$Pr[X > \frac{3}{2}\mu] < e^{-(\frac{1}{2})^2 n/3}$

$= Pr[X > \frac{n}{2}] < e^{-n/36}$

$e^{-n/36} < 1/1000$, solve for n.

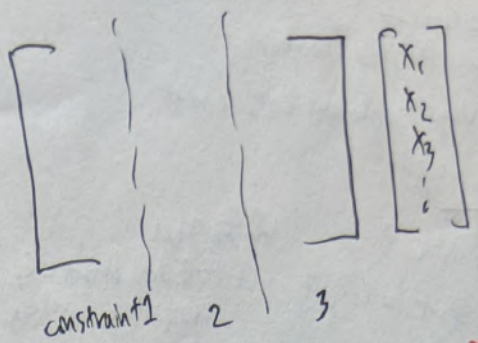
LP

var = $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

m constraints $A\vec{x} \le b$

A is m by n b is m by 1

obj: $\begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$



max Standard form (P)

$\vec{c} \cdot \vec{x}$

$A\vec{x} \le \vec{b}$

$\vec{x} \ge 0$

→ min dual

$\min \vec{b} \cdot \vec{y}$

$A\vec{y} \ge \vec{c}$

$\vec{y} \ge 0$

Want to max a min?

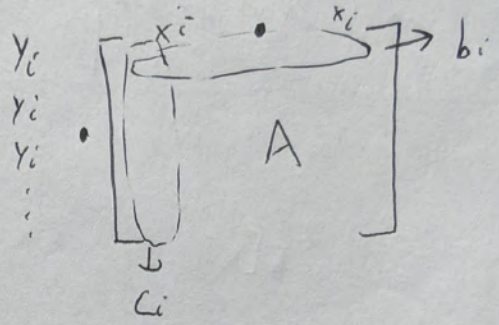
set min = q and max q, ST all $x \ge q$.

- min → max $\vec{c} \to \vec{C}$
- $\ge \to \le$ mult both sides by -1
- $= \to$ use both $\le$ & $\ge$ ②
- $x \in \mathbb{R} \to x \ge 0$ $x_i^+ \ge 0, x_i^- \ge 0$

  $x_i \to (x_i^+ - x_i^-)$

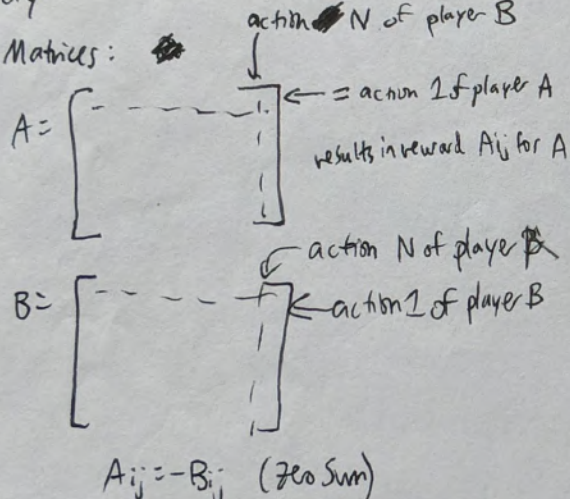
$\max p = \min_D$

- Each $x_i \to i^{th}$ constraint in D
- Each y_i is derived from constraint i in P



Game Theory:

utility Matrices:



$A_{ij} = -B_{ij}$ (zero Sum)

- Often no deterministic algorithm will converge
- But the random will!

  ↳ Nash eq

- V_R max xAy – expected utility of row player if they go first
- V_C min xAy – expected negative utility of column player if they go first

Expect: $V_R \le V_C$

But $V_R = V_C$!

(x^*, y^*) = Nash eq

Nash eq always for 2 person zero sum games

Any game w/ finite players, possible actions does.

Is there Nash eq?

Does E[payoff] remain same despite action chosen?

With some given input, does either player have incentive to change? (select random result and see if reason to change (higher payoff else where?)

oscillation b/w switching actions?

MSTs - Greedy Algs

MST Prop: $G=(V,E)$ is a connected graph w/ cost function. If (u,v) is light edge with U/V, $\exists$ MST w/ (u,v).

Cut - partitions graph

KRUSKALS ALG - $O(E \lg V)$

$T=(V,\emptyset)$

Edges in increasing order:
If edge connects 2 unconnected components: add to T.

Terminate when all vertices in single connected tree.

→ $T \neq \emptyset$

make_set(v) $\forall v$ — $O(|V|) \cdot T_{make\_set}$
sort E by w — $E \log E$
for $e=(u,v) \in E$: — $O(|E|)(T_{Find\_set} + T_{Union})$
  if find_set(v) ≠ find_set(v):
    add e to T:
    UNION(u,v)

$\text{Time} = O(E \log E) + O((V+E)\alpha(V))$
$= O(E \log V)$

$T = O(E \log E) + O(V) \cdot T_{make\_set} + O(E)(T_{Find\_set} + T_{Union})$

Prim's Alg

$s \in V$ start
$T_V = s$
$T_E = \emptyset$
while T not span: → implement Priority Q on vertices, by dist
  Find lowest weight $e=(u,v)$ $s.t.$ $v \notin T_V$
  Add v to T_V
  Add e to T_E

$= O(|E| \cdot T_{Decrease\_key} + |V| \cdot T_{Extract\_min})$
→ Fib heap → $O(|E| + |V| \lg |V|)$

Networks

- $|f|$ = value of flow f = flow going out of s
- $F^* = \max \text{flow} = |f^*|$
- $c(S)$ = capacity of cut S
- $f(S)$ = flow of cut S — $f(S) \leq c(S)$
  - $|f| = f(S)$ for any cut S

$F^* = |f^*| = f^*(S^*) \leq c(S^*)$

Max Flow Min Cut Thm: $F^* = c(S^*)$ → min cut

1. $|f| = c(S)$ for any cut S
2. f is a max flow
3. f has no aug path

G_f = residual network
$c_f(u,v) = c(u,v) - f(u,v)$

If no s-t path in G_f, achieved max flow ✓

Aug Path: path from s→t in G_f
$c_f(p) = \min_{e \in p} c_f(e)$

Edmond-Karp: $O(E^2V)$ ... fewest # of edges

Ford Fulkerson: $O(|E|f)$

- Start w/ 0 flow
- Find aug path via DFS
- Augment flow by pushing $c_f(p)$ along path p
- Repeat until none left

Runtime:
if int: $O(E \cdot V \cdot C)$, capacities $[0, C]$
else: Find max bottleneck path in $O(E \lg V)$
$O(E^2 \lg V \lg VC)$

$|f|$ = value of flow f = flow going out of s

$F^* = \text{max flow} = |f^*|$
value ↳ max flow

G_f = residual network
ST

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

Cap of (u,v) in resid flow = cap (u,v) − flow b/w (u,v) in G

$$|f+f'| = |f| + |f'|$$

$c(S)$ = capacity of cut S
$f(S)$ = flow of cut S $\quad f(S) \le c(S)$
↳ $|f| = f(S)$ for a flow f, any cut S

It follows that $F^* = |f^*|$

$$|f^*| = f^*(S^*)$$
$$f^*(S^*) \le c(S^*)$$
$$F^* = |f^*| = f^*(S^*) \le c(S^*)$$

Augmenting path p → directed $s \to t$ path in G_f

$$c_f(p) = \min_{e \in p} c_f(e)$$

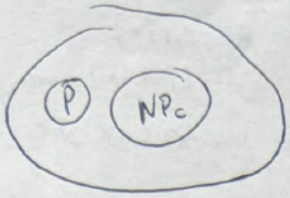
Max Flow Min Cut Thm: $\boxed{F^* = c(S^*)}$ $\quad F^* \le c(S^*)$ bc bottleneck

(1) $|f| = c(S)$ for any $s \to t$ cut S
(2) f is a max flow $F^* = |f|$
(3) f admits no augmenting path

↳ $|f|$ will always = all $f(S)$, but if $|f| = c(S)$ for any cut, then f is a max flow

min cut = cut w/ min $\sum$ of capacities

$P \subseteq NP$



Greedy Interval Scheduling Thm: Select interval that finishes first

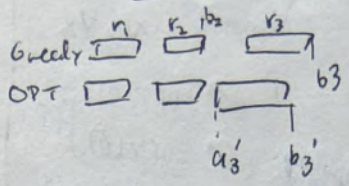
Goal: Find max # of mutually compatible requests

Alg: sort b_i in ascending order

consider each r_i in order – if $a_i > b_j$ last, schedule r_i else discard

$O(n \lg n)$

Pf:



r_3 = 1st interval where Greedy / OPT differ

- $a_3' > b_2$: if not, then Greedy / OPT diff in r_2.
- $b_3' < b_3$ ow Greedy would select b_3'
- $b_3' \geq b_3$ replace r_3' w/ r_3 = OPT'

Greedy still optimal.

Divide & Conquer: problem of size n,

Divide into n/b,

Solve recursively,

Combine solutions of subproblems to get solution of orig. pb

$\Rightarrow T(n) = a \cdot T\left(\frac{n}{b}\right) +$ (combine time)

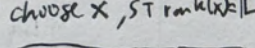
Divide & Conquer App 1: Median Find

Goal: find median of set S of n #s

– rank(x) = # of elems $\leq S \leq x$

Naive – sort, return

Rank Finding → $O(n)$ alg – choose x, ST rank(x) = $|L|+1$

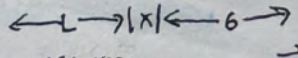


Calc rank x, adjust to L, G if necessary

choosing x:

1. $|S|$ = pow(10)
2. Divide n elem into $n/5$ groups of 5 elem — $\frac{n}{5} \cdot O(1)$
3. Sort, find median in each group
4. Find median of $n/5$ group medians recursively — $T\left(\frac{n}{5}\right)$

4. Find sets L and G ST



$\rightarrow T(|L|)$ or $T(|G|)$

x is $\frac{3}{4}$ balanced.

$T(n) = T\left(\frac{3}{4}n\right) + T\left(\frac{n}{5}\right) + O(n)$

by induction, $T(n) \leq cn$

INTEGER MULT: Div & Conquer

– a, b n bit #s

Goal – Compute $a \cdot b$

Div & Conquer:

$a = 2^{n/2} \cdot x + y$; w, x, y, z $\frac{n}{2}$ bit #s

$b = 2^{n/2} \cdot w + z$

$a \cdot b = 2^n xw + yz + 2^{n/2}(xz + yw)$

Product of $2^{n/2}$ bit #s

– $(x+y)(z+w) = (xz + yw) + xw + yz$

– Subtract to compute $xz + yw$

$T(n) = 3T(n/2) + \Theta(n)$

FFT STRUG

Samples ⟷ Coefficients, best of both worlds conversion

Coeffs → samples

$O(n \lg n)$

poly $A = (a_0, a_1 \ldots a_{n-1})$

$X = \{\vec{x}_0, \vec{x}_1 \ldots \vec{x}_{n-1}\}$

Divide & Conquer

Recursively conquer $\begin{cases} A_{even}(z) \\ A_{odd}(z) \end{cases}$ $A(x) = A_{even}(x^2) + x A_{odd}(x^2)$

Choose set $|X|$ ST it collapses w/ squares

$T(n, |X|) = 2 \cdot T\left(\frac{n}{2}, |X^2|\right) + O(n + |X|)$

$|X^2| = \frac{|X|}{2}$ — use rts of unity

So $T(n, |X|) \in O(n \lg n)$

Samples → coeffs, use IDFT

APPLICATION:

- $A(x), B(x)$ coeffs
- compute $c(x), \forall x$

1) $A^* = DFT(A)$, $B^* = DFT(B)$] FFT

2) $C^* \leftarrow \vec{A^*} \cdot \vec{B^*}$

3) $C = IDFT(C^*)$

Minkowski

$X + Y = \{x + y \mid x \in X, y \in Y\}$

compute $|X+Y|$

- $P_X(x)$ w/ coeff 1 for x^k, $k \in X$,
- Same for P_Y
- P_{X+Y} FFT
- # nonzero coeffs

Amortized Analysis - self organizing List, Competitive Analysis

List L ... Access(x) = Find(x), $\text{rank}_L(x)$

= transpose pair $O(1)$

Goal: sequence of transpositions that minimizes cost $C_A(S)\ \forall S$ in online manner

↳ sequence of keys, S

Worst case = $\Omega(|S| \cdot n)$, last elem every time

MTF: after accessing x, move x to front

① Cost += $\text{rank}_i(x) + \text{rank}_i(x) - 1 \leq 2\,\text{rank}_i(x) - 1$

Comp. Analysis - online alg is α-comp. if $C_A(S) \leq \alpha \cdot C_{OPT}(S) + k$

4 comp.

PF Φ = inversions b/w L_i & L_i^* (MTF, OPT) $\times 2$

- $\Phi(L_0) = 0$
- $\Phi(L_i) \geq 0$
- $\Delta\Phi = \pm 2$ per transposition

L_{i-1}: [A U B |x| C U D] rank = r

L_{i-1}^* = [A U C |x| B U D] r*

→ $r = |A| + |B| + 1$

→ $r^* = |A| + |C| + 1$

MTF step → creates $|A|$ invs, destroys $|B|$ invs

OPT step → t_i creates ≤ 1 inv

→ $\Delta\Phi \leq 2(|A| - |B|) + t_i)$

→ $\hat{C}_i \leq 2r - 1 + 2(|A| - [r - 1 - |A|] + t_i)$ ①

$\leq 4(r^* + t_i)$

$\hat{C}_i \leq 4 C_i^*$

UNION FIND - AMORTIZED ANALYSIS

w/ LL

- Make_set: Create a new set $\{x\}$ $O(1)$
- Find_set: return set w/ x in it $O(1)$
- Union: combine disjoint sets - worst case n^2
  ↳ amortized $n \lg n$, all unions

Forest - data rep is a tree

Make-set $O(1)$
Find-set $O(h)$
Union$(u,v) = O(h_u) + O(h_v)$

- Any time ptrs change for some u, size of set must double. At most $\lg n$ ptrs changed.

Total mops $= O(n \lg n + m)$.

AGGREGATE

$\hat{C} = \frac{\text{Total cost of } k \text{ ops}}{k}$

Accounting

- Table Doubling:
  - $n/2$ elem → $n/2$ coins
  - Use $n/2$ coins to pay for $O(n)$ doubling
  - Total cost $\Rightarrow O(n) - c \cdot \frac{n}{2} = 0$, for good c.
  - Amortized cost/insertion $1 + c = O(1)$

POTENTIAL METHOD

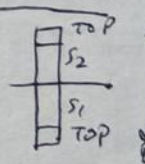
$\hat{C}_i = C_i + \Phi(D_i) - \Phi(D_{i-1})$

actual cost; ↳ keeps state of struct

$\sum_i^n \hat{C}_i = \sum_i^n C_i + \Phi(D_n) - \Phi(D_0)$

Queue of 2 stacks

- ENQUEUE - x @ head of Q
- DEQUEUE - remove elem @ end of tail → move all S_1 to S_2
- PUSH - put x on stack
- POP - remove top elem

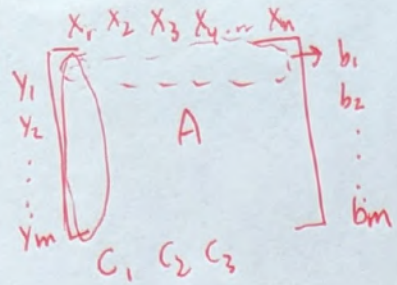


$\Phi(D_i) = 2|S_i|$ ↳ # of obj in stack 1

ENQUEUE $\hat{C}_i = 1 + 2(1) = 3$

DEQ: $\hat{C}_i = 2|S_1^i| + 1 - 2|S_1^i| = 1$

LP



$\max \vec{c} \cdot \vec{x}$

$A\vec{x} \leq \vec{b}$

$\vec{x} \geq 0$

polytope → empty → no solution (infeasible)

unbounded → possibly no best sol.

$\min \vec{b} \cdot \vec{y}$

$A\vec{y} \geq \vec{c}$

Weak: $\vec{c} \cdot \vec{x} \leq \vec{b} \cdot \vec{y}$

$\vec{y} A \vec{x} = \sum_i y_i \left(\sum_j A_{ij} x_j\right) \leq \sum_i y_i b_i \leq \vec{b}\vec{y}$

$\sum_j x_j \sum_i A_{ij} y_i \geq \sum_j x_j c_j = \vec{c}\vec{x}$

$\vec{x} A^T \vec{y} \geq \vec{c} \cdot \vec{x}$

$\vec{c}\vec{x} \leq \vec{b}\vec{y}$

$\vec{c}\vec{x} = \vec{b}\vec{y}$

Game Theory

Utility matrices:

A_{ij} = utility of Player A on outcome (i,j)

B_{ij} = utility of Player B on outcome (i,j)

2 player zero sum = $\forall_{ij} \Rightarrow A_{ij} = -B_{ij}$

∃ Stable outcome? Usu not w/ deterministic strategy
yes w/ random ones ... can there be an improvement in E(utility)

↓

Nash eq

↙

∃ nash eq? Min Max Thm, row goes first

if $V_R := \max_{x \in P} \min_{y \in Q} xAy$ & $V_C := \min_{y \in Q} \max_{x \in P} xAy \Rightarrow V_C = V_R = V$

$\Rightarrow$ Nash eq always exists for 2 player sum games

(any game w/ finite # players, actions)

Ex approx

m machines $M_1, M_2 \ldots M_m$

n jobs $= J_j \subseteq \{1,2,\ldots n\}$ t_i's

$T_j = \sum_{i \in J_j} t_i$

• Lower bound if OPT ≤ APPROX

$\min [\max_i T_i]$

$L = \max\left(\frac{1}{m}\sum_{1 \le i \le n} t_i, \max(t_i)\right)$

↑ Lower bound to optimal

M_L – longest machine

$m \cdot T_L^* \le \sum T_j^* = \sum t_i \le \sum_{\text{all } i} t_i \le m \cdot L$

↓ right before item i

so $T_L = T_i^* + t_i \le L + L \le 2L \le 2OPT$

Decision – Given graph G, and budget k, decide whether $\exists$ X, w/ cost ≤ k. (Y/N)

~~Max~~ search – Given G, k, find a solution X, c ≤ k.

OPT: Given G, find a spanning tree of min weight.

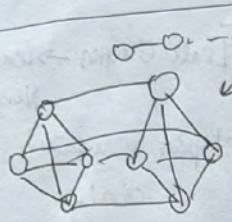
Poly (search) → Poly (decision)
→ poly (opt)

∄ poly (decision) → ∄ poly (either OPT, search)

Ham path – path that visits each vertex once

Ham cycle – whether a cycle of such ↑ exists

NP-hard – every problem P in NP can be reduced to such a problem X (X: NP-hard)



Solving MIS → $O(\lg(V)) = O(\lg(n\Delta))$

Communication = $O(E \lg n)$

↓ messages on every edge

~~Δ~~ rounds

Stationary dists

What will converge?
- any connected, non-bipartite undirected graph
- any lazy connected, undirected graph
- any strongly connected aperiodic graph

What is the dist?

undirected $\pi_v = \frac{d(v)}{\sum_{u \in V} d(u)}$

– directed – solve a few recurrences

Comp Analysis

- aim to prove $\hat{C}_a \le \alpha \cdot C_{OPT} + k$
- use fact that $\hat{C}_a \le C_{actual} + \Delta\Phi$
- rep $\Delta\Phi$ using comparison b/w alg, opt state
- $\hat{C}_a \le$ something

Gradient Descent

$\eta = \frac{1}{\beta}$
$k = \beta/\alpha$

$\vec{x}^{(i+1)} = \vec{x}^{(i)} - \eta_i \vec{\nabla} f(\vec{x}^{(i)})$

$f(\vec{x}+\vec{\delta}) \approx f(\vec{x}) + \underbrace{[\vec{\nabla} f(\vec{x})]^T \delta}_{\text{progress}} + \underbrace{\frac{1}{2}\delta^T \nabla^2 f(\vec{x}) \delta}_{\text{error}}$

let $\delta = -\eta \vec{\nabla} f(\vec{x}) \Rightarrow f(x+\vec{\delta}) \le f(\vec{x}) - \eta \|\vec{\nabla} f(\vec{x})\|^2 + \frac{\beta}{2}\eta^2 \|\vec{\nabla} f(\vec{x})\|^2$

Expected progress $= -\frac{1}{2\beta}\|\vec{\nabla} f(\vec{x}_{(i)})\|^2$

Convex iff $f(\lambda\vec{x} + (1-\lambda)\vec{y}) \le \lambda f(\vec{x}) + (1-\lambda) f(\vec{y})$, $\forall \lambda$, $0 \le \lambda \le 1$, $\vec{x}, \vec{y}$

or $f(\vec{x}+\vec{\delta}) \ge f(\vec{x}) + [\nabla f(\vec{x})]^T \delta$

or $\exists$ a local min (1 local/global)

or if $z^T \nabla^2 f(x) z \ge 0$ or

$\nabla^2 f(x)$ is pos. semidef for all z

Cauchy Schwarz: $z^T x \le \|z\| \|x\|$

β smoothness: $z^T \nabla^2 f(x) z \le \beta \|z\|^2$

α strong convex: $z^T \nabla^2 f(x) z \ge \alpha \|z\|^2$ $\forall x, z$, $\alpha > 0$

$g(x) = f(x^*) + \nabla f(x)^T (x - x^*) + \frac{\beta}{2}\|x - x^*\|^2$, $x^* = \min$

$T = O\left(k \log\left(\frac{f(x_0) - f(x^*)}{\varepsilon}\right)\right)$ to get $f(x^T) - f(x^*) \le \varepsilon$

18.02 Stuff

Stationary points – $\nabla f(x,y) = \vec{0}$

∇^2 = Hessian = $\begin{bmatrix} \partial \nabla f(x,y)/\partial x \to & \\ \partial \nabla f(x,y)/\partial y & \\ \vdots & \end{bmatrix}$

– Plug in pts: if all pos, local min
if indef, saddle pt

$A \le_p B \Rightarrow$ B is at least as hard as A.

Prove P is NP-hard.

3-SAT $\not\le_p$ P

NP-complete:
- ILP
- Circuit-SAT
- SAT / 3-SAT
- CLIQUE
- Set cover
- Hamiltonian-path
- 3-Color

NP-hard:
- Vertex Cover
- Longest path

Vertex Cover

In: Graph $G = (V, E)$

Out: set of vertices $S \subseteq V$ ST $\forall e = (u,v) \in E$
Every edge → one of its endpts is in S

Obj: Min $|S|$, or find one $|S| \le k$

Set cover: Given a set X of n pts, m subsets S_i of X ST $\bigcup_{i=1}^{m} S_i = X$

Find a cover, ST $|C|$ is minimized

Partition: Sorted list of n #s. Find partition of sets A, B ST $\max\{\Sigma A, \Sigma B\}$ is min

Leader Election

Sol: No unique IDs, but numbers
- Choose ID$_i$ from $\{1, 2, \ldots k\}$
- set max$_i$ as max[ID$_i$, incoming msgs]

APPROXES

$\max\left(\frac{C_{APPROX}}{C_{OPT}}, \frac{C_{OPT}}{C_{APPROX}}\right) \le \alpha$

Ex. We want 4-approx – minimize

$\frac{C_{OPT}}{C_{APPROX}} \le 4$, $\frac{C_{APPROX}}{C_{OPT}} \le 4$

Proof: $C_{APPROX} \le 4 C_{OPT}$

MIS

– Leader Election → leader Yes
Neighbor No

– Repeat for unelected

$O(n\Delta)$

Luby's $\theta(\log(n))$

– Random value, send to all neighbors.

– If all received are $< r_i$, join MIS

– If join, announce – all neighbors do not join

– redo for inactive vertices