O( ) f(n) ≥ .

Ω( ) f(n) ≤ .

Θ( ) · ≤ f(n) ≤ .

① Asymptotics

- Relationship between polynomial, exponential, logarithmic time
- Big-Oh notation

(nd)

(e^n)

(log n).

Best-case complexity

average-

worst-. ✓

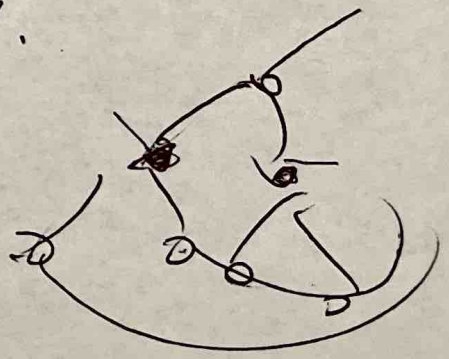
(typical).

(SMP).

Stable matching problem: (no man and woman prefer to be with each other than assigned partner).

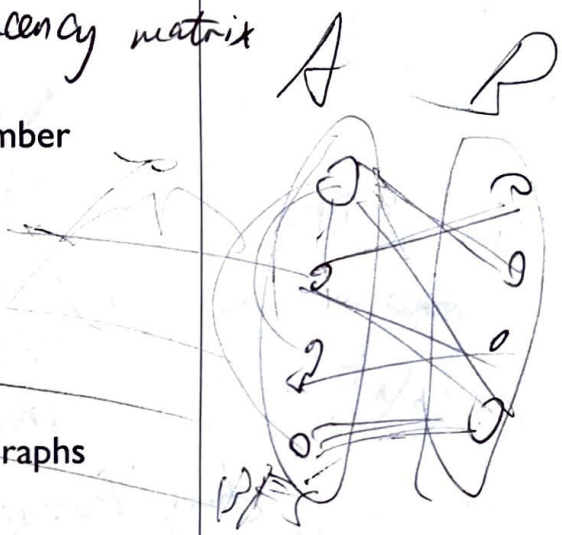
Gale-Shapley algo: $O(n^2)$. propose and reject.

man - optimal.



② Graphs { edge list / adjacency matrix

- Relationship between degree and number of edges
- Cycles, trees
- Graph Search DFT BFT
- Algorithms for coloring
- Negative Edge Weights in Directed Graphs



BFS: queue.

$O(m+n)$

shortest paths

⇓

trees.

structure for. $u \to v$ connectivity

test bipartiteness,

iff (no odd length cycle).

{ odd levels / even levels

two-colorable. $O(m+n)$

$0 \le m \le n(n-1)/2 = O(n^2)$

{ n vertices / m edges.

Sparse: $m \ll n^2$.

$O(n+m) = O(n^2)$

⇓

way better

$m = \sum_{v}^{i} \sum \deg(v)$

$m_{tree} = n-1$.

if $\forall v$ $\deg(v) \ge \frac{n-1}{2}$.

Graph connected.

How to compute?

DAG ⇌ topological order

$O(m+n)$.

DFS: stack.

{ tree edge / back edge. no cross edge.

find articulation point. (vulnerabilities in a network).

Low(v).

③ Greedy Algorithms

- Interval Scheduling
- Approximation algorithms for Vertex Cover, Set Cover
- Minimum Spanning Tree Algorithms, and Cycle and Cut properties
- Union Find-Data Structure

① Greedy stays ahead.

interval scheduling

$O(n \log n)$

optimal.

② Structural.

interval partitioning

Sort by their finishing times

$O(n \log n)$ → pick uncovered edge, add its end points.

→ pick set that minimizes $\frac{\text{cost}}{\#\text{ new elements covered}}$.

$O(k \log n)$

③ exchange arguments.

$O(m \log n)$ with a binary heap.

Prim: priority queue { $O(n^2)$ with an array

MST: Kruskal: union-find data structure

cut property ⇒ MST contains e

cycle property ⇒ MST not contain f.

} by contradiction.

$O(m \log n)$ for sorting

$O(m \log n)$ for union-find.

④ Divide and Conquer Algorithms

- Recurrences (Master Theorem)
- Algorithms for sorting, multiplication, finding closest pairs, matrix multiplication

multiplication { integer, matrix, polynomials.

$T(n) = aT(n/b) + cn^k$. $(n>b)$.

$= a(aT(n/b^2) + c(n/b)^k) + cn^k$. unbalanced division less good.

$= a^{\log_b n} + \beta n^k \log_b n$.

$= n^{\log_b a} + \beta n^k \log_b n$

[$\log_b a$ vs k].

$T(n) = 8T(n/2) + 4(n/2)^2$

$= 8T(n/2) + n^2$.

$\log_b a = \log_2 8 = 3$

$k = 2$.

$\Rightarrow T(n) = O(n^3)$

Strassen's: $T(n) = 7T(n/2) + cn^2$

sub problem disjoint

integer multiplication

$T(n) = 4T(n/2) + \Theta(n)$

↓ add, shift.

quick sort

merge sort: $T(n) = 2T(n/2) + cn$.

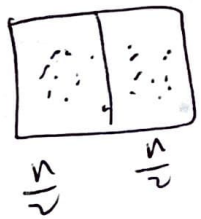
must be adjacent

$= 2\left(2T(n/4) + c(n/2)\right) + cn$.

$= \cdots \log(n)\, T(1) + n\log(n)$

1-D version: $O(n \log n)$ (sort).

2-D version: divide + conquer + combine.



$D(n) \le \begin{cases} 0 & n=1 \\ 2D(n/2) + n & n>1 \end{cases} \Rightarrow n\log(n)$.

⇒

OPT = optimal solution

[utility, constraint].

D&C: independent sub-problems

DP: overlapping sub-problems.

⑤ Dynamic Programming

Top-down vs bottom-up.

Express OPT in terms of OPT for smaller problems.

- Design the recurrence using subproblems, then write the program
- Algorithms for sequencing related problems, edit distance, knapsack, weighted interval scheduling. (find max weight subset of mutually compatible jobs).

$O(nW)$.

$$OPT(j) = \begin{cases} 0 & j=0 \\ \max\{v_j + OPT(p(j)),\ OPT(j-1)\} & \text{otherwise.} \end{cases}$$

Main loop is $O(n)$

Sorting. $O(n \log n)$

Bellman-Ford

$$OPT(i,j) = \begin{cases} \delta & \text{if } i=0 \\ \min\{\alpha_{x_i y_j} + OPT(i-1,j-1),\ \delta + OPT(i-1,j),\ \delta + OPT(i,j-1)\} & \text{otherwise.} \\ j\delta & \text{if } j=0. \end{cases}$$

pre-determined

$$OPT(i,w) = \begin{cases} 0 & i=1,\dots,n \\ OPT(i-1,w) & \\ \max\{OPT(i-1,w),\ v_i + OPT(i-1, w-w_i)\} & \end{cases}$$

(weight limit w).

$O(nW)$.

All pairs shortest paths:

- early: $O(mn^3)$
- dp: $O(n^3)$

Travel Salesman problem:

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

$$T(S,k) = \min_{k' \in S,\ k' \neq k} \{T(S - \{k\}, k') + d_{k',k}\}.$$

formulate as min-cut.

add { source: foreground
sink: background.

directed graph, no parallel edges.

⑥ Network Flows

- Algorithms for max-flow, min-cut
- Applications to Image segmentation, project selection, edge-disjoint paths, maximum matchings in bipartite graphs, Hall's theorem.

capacity $= \sum_{e \text{ out of } A} c(e)$.

$v(f)$ = flow. value of flow $= \sum_{e \text{ out of } s} f(e)$

min-cut: find s-t cut of minimum capacity.

max-flow: find s-t flow of maximum value.

max number edge-disjoint s-t paths equals max flow value (unit capacity)

no edge in common.

Weak duality: $\forall A$.

$$v(f) \le \sum_{e \text{ out of } A} c(e)$$

Marriage theorem: Let $G = (L \cup R, E)$ be a bipartite graph. with $|L| = |R|$. Then, G has a perfect matching iff. $|N(S)| \ge |S|$ for all subsets $S \subseteq L$.

max-flow algo:

greedy: augment until stuck. ⇒ not optimal.

residual graph: $G_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E. \\ f(e) & \text{if } e^R \in E. \end{cases}$

⇓

$O(mn)$.

edge vertex.

Ford-Fulkerson Algo..

find path ← exponential number of augmentations

update residual graph

until no path

{ flow f is a max flow iff there are no augmenting paths.

☆ { The value of the max flow is equal to the value of the min cut.

min vertex cover

max. edge matching

max matching in G = max flow in G'

⑦ Linear Programming

- Minimax Theorems
- Duality
- Strong Duality Theorem

an optimal solution exists ①

weak: $c \cdot x \le yAx \le y \cdot b$

$c \cdot x^* = y^* A x^* = y^* \cdot b$

$\max \; c \cdot x$
$s.t. \; Ax \le b$
$x \ge 0$

$\Longleftrightarrow$

$\min \; y \cdot b$
$s.t. \; yA \ge c$
$y \ge 0.$

$\max \; c \cdot x$
$s.t. \; Ax \le b$
$x \ge 0$

$\Longleftrightarrow$

$\max \; -b^T \cdot y^T$
$s.t. \; -A^T y^T \le -c$
$y^T \ge 0$

$b = c^T$

$A = -A^T$

$\min_y \max_x yAx =$

$\max_x \min_y yAx$

It doesn't matter whether the row-player or the column player has to commit to a strategy first.

⑧ Randomized Algorithms

- Linearity of Expectation $\mathbb{E}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} \mathbb{E}(X_i)$
- Algorithms for Min-Cut, Dominating Set, Flows, Testing when a polynomial is 0.

$\det(B) =$

$\sum_{\text{permutations } \sigma:[n]\to[n]} \prod_{i=1}^{n} A_{i\sigma(i)}$

"Laplace expansion"

recursive call.

Hot edges that cross from A to B is minimized.

either contained in S or is a neighbor of S

the Pr that k edges of the min-cut are never picked is at least.

$\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{2}{3}\right) = \frac{2}{n(n-1)}$

Let $p(x_1,\ldots x_n) \neq 0$ be a polynomial of degree d. Let S be a finite set of numbers. Then if $x_1,\ldots,x_n$ are sampled uniformly from S.

$Pr[p(x_1,\ldots x_n)=0] \leq d/|S|$

(Schwartz-Zippel).

decision problems:
problems with "yes" or "no" answers

⑨ NP : Nondeterministic polynomial-time.

- Definition of NP : decision problems for which there exists a poly-time certifier.
- NP-completeness
- NP-completeness for 3SAT, Independent Set, 3-coloring

poly-time : $p(|x|)$ decision steps
where $p(\cdot)$ is some poly.

P: $p(\cdot)$ algo.
NP: $p(\cdot)$ certifier (check ....)
EXP: $exp(\cdot)$ algo

$P \subseteq NP \subseteq EXP$

$P = NP$? : is the decision problem as easy as the certification problem?

2-Coloring : BFT

↘ a set of vertices that do not contain any edges.

~~for every X, poly $\Rightarrow P$~~
~~for some X, poly $\Rightarrow NP$~~

NP-Complete : a problem Y (in NP) is NP-complete if for every problem X in NP, $X \leq_p Y$.

$\Rightarrow$ if X NP-complete ($\forall W$ in NP, $W \leq_p X$) and Y is a problem in NP with the property that $X \leq_p Y$, then Y is NP-complete.

(Problem X polynomial reduces to problem Y ($X \leq_p Y$) if arbitrary instances of problem X can be solved using { Polynomial number of standard computational steps, + Poly... number of calls to subroutine that solves problem Y.