L03 Greedy Algorithms

Greedy Analysis Strategies:

- *Greedy algorithm stays ahead.* Show that after each step of the greedy algorithm, its solution is at least as good as any other algorithm's.
- *Exchange argument.* Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.

① Interval Scheduling

Consider jobs in increasing order of finish time.

② Scheduling to Minimize Lateness

Earliest deadline first.

③ Optimal Caching

Farthest-in-future. Evict item in the cache that is not requested until farthest in the future.

④ Clustering

Single-link k-clustering algorithm.

L04 Divide and Conquer

设 $a \geq 1$ 和 $b > 1$ 为常数，设 f(n)为一函数，T(n)由递归式

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

其中$\frac{n}{b}$指$\left\lfloor\frac{n}{b}\right\rfloor$和$\left\lceil\frac{n}{b}\right\rceil$，可以证明，略去上下去整不会对结果造成影响。那么 T(n)可能有如下的渐进界

(1)若 $f(n) < n^{\log_b^a}$，且是多项式的小于。即

$\exists\, \varepsilon > 0$，有 $f(n) = O\left(n^{\log_b^a - \varepsilon}\right)$，则 $T(n) = \Theta\left(n^{\log_b^a}\right)$

(2)若 $f(n) = n^{\log_b^a}$，则 $T(n) = \Theta\left(n^{\log_b^a} \log n\right)$

(3)若 $f(n) > n^{\log_b^a}$，且是多项式的大于。即

$\exists\, \varepsilon > 0$，有 $f(n) = \Omega\left(n^{\log_b^a + \varepsilon}\right)$，且对$\forall\, c < 1$ 与所有足够大的 n，有 $af\left(\frac{n}{b}\right) \leq cf(n)$，则 $T(n) = \Theta(f(n))$

① Merge sort

Def. T(n) = number of comparisons to mergesort an input of size n.

$$T(n) = \begin{cases} 0 & if\ \ n = 1 \\ T\left(\left\lceil\frac{n}{2}\right\rceil\right) + T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n & otherwise \end{cases}$$

$T(n) = O(n \log_2 n)$.

② Closest Pair of Points

$$T(n) \leq 2T(n/2) + O(n) \to T(n) = O(n\log n)$$

③ Integer Multiplication

Divide each n-digit integer into two ½n-digit integers (Karatsuba-Ofman, 1962)

$$xy = 2^n \cdot x_1 y_1 + 2^{\frac{n}{2}} \cdot ((x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0) + x_0 y_0$$

$$T(n) \leq T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + T\left(\left\lceil\frac{n}{2}\right\rceil\right) + T\left(1 + \left\lceil\frac{n}{2}\right\rceil\right) + \Theta(n)$$

$$T(n) = O\left(n^{\log_2 3}\right) = O\left(n^{1.585}\right)$$

④ Matrix Multiplication

Divide each n-by-n matrix into four ½n-by-½n blocks

7 multiplications, 18 additions.

$$T(n) \leq 7T\left(\frac{n}{2}\right) + \Theta(n^2) \to T(n) = \Theta\left(n^{\log_2 7}\right) = O\left(n^{2.81}\right)$$

Best known. $O(n^{2.3728596})$ [Alman & Williams, 2020]

Conjecture. $O(n^{2+\epsilon})$ for any $\epsilon > 0$.

⑤ Convolution and FFT

$$A(x) = A_{even}(x^2) + xA_{odd}(x^2).$$
$$A(-x) = A_{even}(x^2) - xA_{odd}(x^2).$$

Combine.

$$A(\omega^k) = A_{even}(v^k) + \omega^k A_{odd}(v^k), 0 \leq k < n/2$$
$$A(\omega^{k+n/2}) = A_{even}(v^k) - \omega^k A_{odd}(v^k), 0 \leq k < n/2$$

Integer multiplication

Convert to binary polynomial, then multiply.

O(n log n) complex arithmetic steps.

L05 Dynamic Programming

Top-down: May skip unnecessary sub-problems

Bottom-up: Save the overhead in recursion

① Weighted Interval Scheduling - $O(n \log n)$ [O(n) if pre-sorted start & finish]

OPT(j) = value of optimal solution to the problem consisting of job requests 1, 2, ..., j.

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

② Knapsack Problem - $\Theta(nW)$

Def. OPT(i, w) = max profit subset of items 1, …, i with weight limit w.

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

③ RNA Secondary Structure – $O(n^3)$

OPT(i, j) = maximum number of base pairs in a secondary structure of the substring $b_i b_{i+1} \dots b_j$.

$$OPT(i,j) = \begin{cases} 0 & i \geq j-4 \\ \max(OPT(i,j-1), 1 + \max_t\{OPT(i,t-1) + OPT(t+1,j-1)\}) & i < j-4 \end{cases}$$

④ Sequence Alignment - $\Theta(mn)$ time and space

Edit distance. Gap penalty δ; mismatch penalty α_{pq}.

Def. OPT(i, j) = min cost of aligning strings $x_1 x_2 \dots x_i$ and $y_1 y_2 \dots y_j$.

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

⑤ Sequence Alignment in Linear Space

-$O(mn)$ time to compute $f(\bullet, n/2)$ and $g(\bullet, n/2)$ and find index q.

- $T(q, n/2) + T(m-q, n/2)$ time for two recursive calls.

$$T(m,n) \leq cmn + T(q,n/2) + T(m-q,n/2)$$

⑥ Shortest Paths

Def. OPT(i, v) = length of shortest v-t path P using at most i edges.

$$OPT(i,v) = \begin{cases} \infty & if\ \ i = 0, v \neq t \\ 0 & if\ \ v = t \\ \min\left\{OPT(i-1,v), \min_{(v,w)\in E}\{OPT(i-1,w) + c_{vw}\}\right\} & otherwise \end{cases}$$

if no negative cycles, then OPT(n-1, v) = length of shortest v-t path.

$\Theta(mn)$ time, $\Theta(n^2)$ space.

O(n) extra space, $O(mn)$ time

⑦ Distance Vector Protocol

Bellman-Ford "Routing by rumor." each router performs n separate computations, one for each potential destination node. "counting to infinity" Each router also stores the entire path. Requires significantly more storage.

⑧ Negative Cycles in a Graph

Can detect negative cost cycle in O(mn) time. Add new node t and connect all nodes to t with 0-cost edge. Check if OPT(n, v) = OPT(n-1, v) for all nodes v.

- if yes, then no negative cycles
- if no, then extract cycle from shortest path from v to t.

L06 Network Flow

① Residual Graph, Augmenting Path, Ford-Fulkerson Algorithm

Def. The capacity of a cut (A, B) is: $cap(A,b) = \sum_{e\ out\ of\ A} c(e)$.

Capacity Scaling: $\mathbf{O(m^2 \log C)}$ time.

② Bipartite Matching

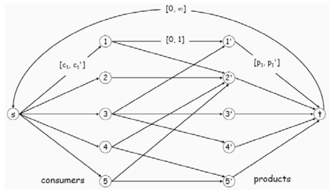
Perfect Matching: $|L| = |R|$, G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

③ Extensions to Max Flow

Circulation with Demands. Add new source s and sink t. For each v with d(v) < 0, add edge (s, v) with capacity -d(v). For each v with d(v) > 0, add edge (v, t) with capacity d(v).

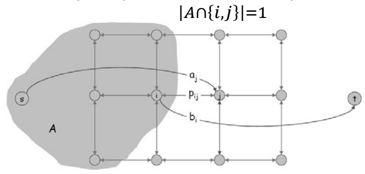
④ Survey Design

Integer circulation = feasible survey design.



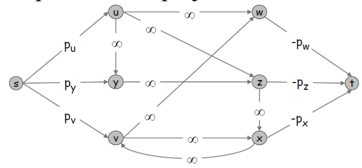
⑤ Image Segmentation

Find partition (A, B) that maximizes: $\sum_{i\in A} a_i + \sum_{j\in B} b_j - \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij}$



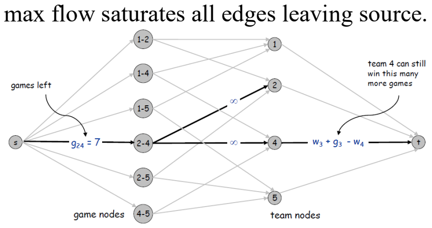
⑥ Project Selection

Min cut formulation. (A, B) is min cut iff $A - \{s\}$ is optimal set of projects.



⑦ Baseball Elimination

Assume team 3 wins all remaining games. $w_3 + g_3$ wins. Team 3 is not eliminated iff max flow saturates all edges leaving source.



Explanation for Sports Writers

Team z is eliminated iff there exists a subset T* such that $\frac{w(T^*)+g(T^*)}{|T^*|} > w_z + g_z$.

Define T* = team nodes on source side of min cut. Observe $x - y \in A$ iff both $x \in T^*$ and $y \in T^*$. $g(S - \{z\}) > cap(A,B)$.

L07 NP and Computational Intractability

① Polynomial-Time Reductions

Problem X polynomial-time *reduces to* problem Y if arbitrary instances of problem X can be solved using polynomial number of standard computational steps, plus polynomial number of calls to oracle that solves problem Y. (not reduce from)

② NP stands for nondeterministic polynomial-time.

P. Decision problems for which there is a poly-time algorithm.

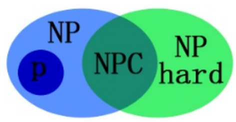
EXP. Decision problems for which there is an exponential-time algorithm.

NP. Decision problems for which there is a poly-time certifier.

③ Terminology

NP-complete. A problem in NP such that every problem in NP polynomial reduces to it. **NP-hard.** A problem such that every problem in NP reduces to it.

co-NP.



Complements of decision problems in NP. for *no* instance, there is a succinct disqualifier. (e.g. TAUTOLOGY, NO-HAM-CYCLE, PRIMES.)

- If NP≠co-NP, then P≠NP.

- P ⊆ NP ∩ co-NP.
- Factoring is in NP∩co-NP, but not known to be in P. (Factor: Given two integers x and y, does x have a nontrivial factor less than y?)

③ NP-complete.
Step 1. Show that Y is in NP. (polytime cert)
Step 2. Choose an NP-complete problem X.
Step 3. Prove that $X \leq_p Y$.

CIRCUIT-SAT. Given a combinational circuit built out of AND, OR, and NOT gates, is there a way to set the circuit inputs so that the output is 1?
3-SAT. Given CNF formula Φ, each clause contains exactly 3 literals, does it have a satisfying truth assignment?
INDEPENDENT SET. Given a graph G = (V, E) and an integer k, is there a subset of verticesS ⊆ Vsuch that $|S| \geq k$, and for each edge at most one of its endpoints is in S?
VERTEX COVER. Given a graph G = (V, E) and an integer k, is there a subset of verticesS ⊆ Vsuch that $|S| \geq k$, and for each edge, at least one of its endpoints is in S?
SET COVER. Given a set U of elements, a collection $S_1, S_2, \dots, S_m$ of subsets of U, and an integer k, does there exist a collection of ≤ k of these sets whose union is equal to U?
HAM-CYCLE. given an undirected graph G = (V, E), does there exist a simple cycle Γ that contains every node in V.
DIR-HAM-CYCLE. given a digraph G = (V, E), does there exists a simple directed cycle Γ that contains every node in V?
TSP. Given a set of n cities and a pairwise distance function $d(u, v)$, is there a tour of length ≤ D?
LONGEST-PATH. Given a digraph G = (V, E), does there exists a simple path of length at least k edges?
3D-MATCHING. Given disjoint sets X, Y, and Z, each of size n and a set $T \subseteq X \times Y \times Z$ of triples, does there exist a set of n triples in T such that each element of $X \cup Y \cup Z$ is in exactly one of these triples?
3-COLOR. Given an undirected graph G does there exist a way to color the nodes red, green, and blue so that no adjacent nodes have the same color?
k-REGISTER-ALLOCATION. Assign program variables to machine register so that no more than k registers are used and no two program variables that are needed at the same time are assigned to the same register.
SUBSET-SUM. Given natural numbers $w_1, \dots, w_n$ and an integer W, is there a subset that adds up to exactly W?
PARTITION. Given natural numbers $v_1, \dots, v_m$, can they be partitioned into two subsets that add up to the same value?
SCHEDULE-RELEASE-TIMES. Given a set of n jobs with processing time t_i, release time r_i, and deadline d_i, is it possible to schedule all jobs on a single machine such that job i is processed with a contiguous slot of t_i time units in the interval $[r_i, d_i]$?

L08 PSPACE
① Quantified satisfiability: Let $\Phi(x_1, \dots, x_n)$ be a boolean CNF formula. Is the following propositional formula true?
$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \dots \forall x_{n-1} \exists x_n \ \Phi(x_1, \dots, x_n)$
Q-SAT ∈ PSPACE-complete
PLANNING ∈ PSPACE

Competitive facility location ∈ PSPACE-complete
Input. Graph G = (V, E) with positive edge weights, and target B.
Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors has been selected.
Can second player guarantee at least B units of profit?

L09 Extending tractability
① Vertex Cover: small k – pick an edge, remove any vertex of the two, # of edges ≤ k(n − 1). $O(2^k kn)$
② Independent set on trees: greedy pick leaf $O(n)$; Weighted: DP. w/,w/o root, $O(n)$.
③ Circular arc coloring: DP, $O(k!\,n)$. List all possibilities. For small k.
④ Vertex cover in bipartite graphs: max matching = min vertex cover. Network flow.

L10 Local Search
① Gradient descent: vertex cover, remove dots.
② Metropolis algorithm: update with prob $e^{-\Delta E/(kT)}$ $(\Delta E > 0)$. Simulated annealing.
③ Hopfield neural networks: flip bad nodes. Progress $\Phi(S) = \sum_{e\ good} |w_e|$. $W = \sum_e |w_e|$.
④ Maximum cut: Single-flip neighborhood, greedy. Big-improvement-flip: flip increase ≥ $\frac{2\epsilon}{n} w(A, B)$ => $(2+\epsilon) w(A, B) \geq w(A^*, B^*)$. $O(\epsilon^{-1} n \log \sum_e w_e)$ flips; KL-neighborhood.
⑤ Nash equilibria.

L11 Lower Bounds
① If an algorithm takes too little time, it must sometimes produce the wrong answer.
② Merge two list: 2n-1; Max: n-1; Max-min: 3n/2-2; Sort: $\Omega(n \log n)$

L12-15 Randomized algorithms
① Max-cut, Monte Carlo, random put, in expectation 2-approximation.
② Quicksort, random pivot.
③ Hash table. Closed/open addressing. Load factor α. Universal hashing: $\Pr_{h \in H}[h(x) = h(y)] = 1/m$. Perfect hashing.
④ Bloom filters. $\left(1 - \frac{1}{m}\right)^{nk} \approx e^{-\frac{nk}{m}}$.
⑤ Fingerprint. $O(\log n)$bits, $O(1/n)$ FP.
⑥ String matching. Monte Carlo, O(n+m), FPprob O(1/n). Las Vegas, expected O(m+n).
⑦ **Union Bound.** If $Pr[E_i] = p_i$ $(1 \leq i \leq k)$, then $\Pr[E_1 \cup \dots \cup E_k] \leq p_1 + \dots + p_n$.
⑧ **Markov's Inequality.** Positive r.v. X: $\Pr[X \geq a] \leq E[X]/a$ $(a > 0)$.
⑨ **Chebyshev's Inequality.**
$\Pr[|X - E[X]| \geq a] \leq Var[X]/a^2$ $(a > 0)$.
⑩ **Chernoff Bounds.**
(1) Let $X_1, X_2, \dots, X_n$ be independent r.v.s, with values in {0,1} s.t. $E[X_i] = p_i$ for all i. Let $X = \sum_i X_i$ and $\mu = E[X] = \sum_i p_i$.Then
For $0 < \delta \leq 1, \Pr[X \geq (1+\delta)\mu] \leq e^{-\mu\delta^2/3}$.
For $\delta > 1, \Pr[X \geq (1+\delta)\mu] \leq e^{-\mu\delta \ln \delta/3}$.
For $0 \leq \delta \leq 1, \Pr[X \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2}$.
(2) Same setting as above. For any $\delta > 0$,

$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$$

$$\Pr[X < (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^\mu$$

(3) Let $X_1, X_2, \dots, X_n$ be independent {-1, 1} valued r.v.s, with $Pr[X_i = 1] = Pr[X_i = -1] = 1/2$ for all i. Let $X = \sum_i X_i$. Then for any $\delta \geq 0$,

$$Pr[X \geq \delta] = Pr[X \leq -\delta] \leq e^{-\frac{\delta^2}{2n}}$$

(4) Two-sided Chernoff Bound
Let $X_1, X_2, \dots, X_n$ be independent r.v.s, $0 \leq X_i \leq 1$. Let $X = \sum_i X_i$ and $\mu = E[X] = \sum_i E[X_i]$. Then for any $\epsilon > 0$,

$$Pr[|X - \mu| \geq \epsilon\mu] \leq 2\exp(-\frac{\epsilon^2}{2+\epsilon}\mu)$$

$(2 + \epsilon \ can\ be\ replaced\ with\ 3.)$

More applications: Load balancing ($m = 16n \ln n$), Set balancing ($\sqrt{4m \ln n}$?), 2D LP (O(n)), d-D LP (O(d!n)) …

L15-17 Approximation algorithms
① Set covering. $\ln(n)$-approximation. Pick the cheapest.
② Makespan scheduling. NPC. List scheduling: 2-approximation. Longest processing time (LPT) schedule: 4/3-approximation.
③ The knapsack problem. polynomial time approximation scheme (PTAS), $(1+\epsilon)$-approximation. Run time $O(n^3/\epsilon)$. Scaling factor $\theta = \epsilon v^*/2n$.
④ Vertex cover: 2-approximation. Weighted vertex cover: Integer linear programming

$$\begin{array}{llrcll} (ILP) & \min & \sum_{i \in V} w_i x_i & & & \\ & \text{s.t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\ & & x_i & \in & \{0, 1\} & i \in V \end{array}$$

$$\begin{array}{llrcll} (LP) & \min & \sum_{i \in V} w_i x_i & & & \\ & \text{s.t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\ & & x_i & \geq & 0 & i \in V \end{array}$$

$S = \{i \in V : x_i^* \geq 1/2\}$. 2-approximation.
⑤ Traveling Salesman Problem (TSP)
Metric TSP (triangle inequality):
2-approximation, minimum spanning tree + DFS; 1.5-approximation, MST + odd degree perfect matching + Euler path + short cut.
⑥ k-Center problem (triangle inequality)
NPC, Gonzalez's algorithm, 2-approximation. Add center to farthest site.