ALGO FINAL NOTES

Intro: $f(n) = \Theta(g(n))$: $\lim_{n \to \infty} \frac{f(n)}{g(n)}$ $\infty \to \Omega(n)$, $(0, \infty) \to \Theta(n)$, $0 \to O(n)$

$\Theta(g(n)) = \{ f(n) :$ there exist c_1, c_2, n_0 st $0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$ for $n > n_0 \}$

Master's theorem: $T(n) \le a\,T(n/b) + O(n^d)$

$a < b^d \to O(n^d)$ good

$a = b^d \to O(n^d \lg n)$ ok

$a > b^d \to O(n^{\log_b a})$ bad

Randomized Algos: for $\vec{d} \neq 0$, $\text{prob}_{\text{rand } \vec{v}} [\vec{d} \cdot \vec{v} = 0] \le \frac{1}{2}$

$H = \{h_1, h_2, \dots\}$ is a UHF if $\text{Prob}_{h \in_R H} [h(x) = h(y)] \le \frac{1}{m}$ so $\sum E(C_{xy}) \le \binom{n}{2}\frac{1}{m} \le \frac{n^2}{2m}$

Graphs: DFS(G, σ):
    visited, first, last, time, F
    for v in σ:
        if v not visited: DFS(G, v)

DFS(G, v):
    visited[v] = true, time += 1, first[v] = time
    for u in v.neighbors:
        if u not visited: add (u, v) to F, DFS(G, u)
    time += 1, last[v] = time

Forest: $u, v \in F$
back: $u, v \in E \setminus F$, u descends v in F
forward: $u, v \in E \setminus F$, v descends u in F
cross: remaining $\in E$

nested interval: v after u, either [[ ]] or [ ] [ ]
edge property: $u, v \in E$ and u visited first, [[ ]]
path property: $u_1, u_2, \dots, u_k$ and u_1 first, [[ ]]
DAG: decreasing last[i] is topological ordering
Finding cycles: run DFS any σ, check if u, v is back edge
SCC: run DFS any order, σ = decreasing last(v), H = reverse(G), run DFS(H, σ) & return components

Paths: DIJKSTRA(G, s): $O(m + n\log n)$
    dist, parent, Q has s, visited
    while Q not empty:
        v = Q.remove ; visited.add(v)
        for u in v.neighbors:
            if dist[v] + c(v,u) < dist[u]:
                dist[u] = dist[v] + c(v,u)
                parent[u] = v
        add u with smallest dist to Q

LBSW(G, s, k): $O(km)$
    for i = 1 to k:
        for $v \in V$:
            $d[v, i] = \min \begin{bmatrix} d[v, i-1] \\ d[u, i-1] + c(u,v) \\ \text{for } u \text{ s.t. } u,v \in E \end{bmatrix}$

BELLMAN-FORD(G, s):
    run LBSW(G, s, n): $O(nm)$
    if v s.t. dist[v, n] < dist[v, n-1]:
        return cycle
    else: return shortest path tree

Flows: def Flow f, val(f), s,t-cut, $\partial^+ S$, residual network
ford-fulkerson: find augmenting path, augment it.
    running time $O(m+n)(nU) = O(mnU)$
    $\underbrace{O(m+n)}_{\text{find any path}}\underbrace{(nU)}_{\text{\# loops (max flow)}}$

Hall's theorem: perfect matching $\to$ for any $S \subseteq$ left, |neighbors of S in R| $\ge$ |S|

ALGO PROBLEMS

DIVIDE & CONQUER:

merge-sort: combine with 2 pointers $O(n \lg n)$ — class

counting inversions: cross-inversions in sorted lists takes $O(n)$ → total $O(n \lg^2 n)$ → wires on chips

~~max-range subarray~~: find best soln between left & right $O(n \lg n)$ $O(n)$ time

multiplying polynomials: $B = (p_1+p_2)(q_1+q_2) - A - C$ trick $O(n^{\log_2 3})$

closest pair of points: use geometry to merge in linear time $O(n \lg n)$

$A[i] = i$ on sorted array: recurse on left half if $A[m] > m$ — UGP

local minima: check midpoint & go to half w/ smaller element — Pset

row maxima in criss-cross matrix: use criss-cross property after getting middle

maximal points: combine G & R in linear time by removing dominated points

majority: find majority of halves and combine

DP:

subset sum w/ repetitions: $F(m,b) = \max(F(m-1,b), F(m-1, b - z_m a_m)$ for $1 \le z_m \le b/a_m)$ 0/1 — class

knapsack: $F(m,b) = \max(F(m-1,b), F(m-1, b-w_m)+p_m)$ ①

LCS: $LCS(i,j) = \max\{LCS(i-1,j), LCS(i,j-1), LCS(i-1,j-1) + 1_{ij}\}$ ②

coffeeshops: $F(i) = \max(F(i-1), F(j)+p_i)$ ③ — Pset

~~word sequences~~: $F(m) = \max_{0 \le i \le m-1} (F(i)$ and $D(S[i+1:m]))$ ④

longest palindromic subsequence: $F(i,j) = \max(F(i,j-1), F(i+1,j-1), F(i+1,j-1) + 2_{ij})$ ②

pebbling a checkerboard: $F(i,m) = colVal(i,m) + \max_{t \in compatPlacements(i)} F(t, m-1)$

counting heads: $P(m,l) = P(m-1,l)\cdot(1-p_m) + P(m-1,l-1)\cdot p_m$ ⑤ — UGP

knapsack with budget: $F(m,b,c) = \max(F(m-1,b,c), F(m-1, b-w_m, c-c_m)+p_m)$ ①

max range subarray: $F(i) = \max(F(i-1), A[i] - \min A[1:i])$

weighted interval packing: $F(i) = \max(F(i-1), F(prev(i)) + w_i)$ ③

~~longest increasing subsequence~~: $LIS(m) = 1 + \max_{1 \le j < m:\, A[j] \le A[m]} LIS(j)$ ④

~~string splitting~~: $F(i,j) = |a_j - a_i| + \min_{i<k<j} F(i,k) + F(k,j)$ ④

boosting reliability: $F(m,b) = \max_{1 \le k_m \le b/c_m} F(m-1, b - c_m k_m)\cdot p(k_m)$ ⑤

~~opening gyms~~: $F(l,m) = \min_{1 \le i \le m} F(l-1,i) + \sum_{i \le j \le m} |d_j - d_{mid}|$ ①

Randomized Algos:

check 2 n-bit vectors are equal w/ minimum comm: check dot-product with random string — Pset

k^{th} smallest element: $O(n)$ running time by finding middle-ish pivot and recurse appropriately

heavy hitters: need to review — UGP

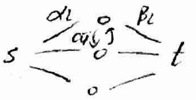
flipping coins for 1/6 prob

Pset

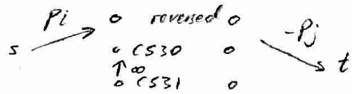
Graphs: pouring water puzzle: vertex representing states, and search
terms to take CS curriculum: sort DAG in topological order and then traverse
2SAT. construct problem. if x and $\bar{x}$ in SCC then not possible

Paths & Cycles: min cost cycle w/ positive edges: Dijkstra on every vertex, minimize cost $s \to t + t \to s$
undirected: Dijkstra on every $G-(u,v)$. minimize cost $(u,v) + v \to u$
modify dijkstra to find max capacity path
all pairs of shortest paths: naive: Bellman Ford n-times $\to n \cdot O(mn)$
$dist[i,j,k] = \min(dist(i,j,k-1), dist(i,j,k-1) + dist(k,j,k-1))$
up to k vertices used

Flow: $v_{in} \xrightarrow{cap(x)} v_{out}$



Fenyman: $val(i,j) = T_{ij} M_0 - P_{ij} - \varepsilon$ and use Bellman-Ford to find cycle
Training wizards: vertex for station, subgraph for system.
Project matching:



UGP6: reverse graph, walk $\to$ path
priority queue: insert/modify in $\log(n)$ time

UGP7: bipartite testing: BFS, if all $d[u] \neq d[v]$ for (u,v), then bipartite
verify shortest path tree:

SUPP DIVIDE & CONQUER: k^{th} smallest element in sorted A & B: if $A[k/2] < B[k/2] \to$ FIND: $A[k/2+1:n], B[1:k/2+1], k-k/2]$