GREEDY ALGORITHMS

kruskal: repeatedly add the next lightest edge to G' that does not create a cycle. makeset(x), find(x): identify x's connected component (labeled by root), union(x,y): merge connected components. For all $x \in V$: $makeset(x)$. X={} (empty set). for all $e = (x, y)$ (in increasing order of weight), if $find(x) \neq find(y) : X = X \cup \{e\}, union(x, y)$. Cost = sorting edges + $2 \cdot |E|$·cost(find) + $(|V| - 1)$·cost(union) + $|V|$·cost(makeset).

prim: similar to Dijkstra. Start form any vertex, find the lightest edge extending from your current tree and added that tree and ending vertex.

horn formulas: $(x_1 \wedge x_2 \wedge \cdots \wedge x_k) \Rightarrow x_{k+1}$ OR $(\bar{x_1} \vee \bar{x_2} \vee \cdots \vee \bar{x_k} \vee x_{k+1})$. Start with all vars = false. While there is an unsatisfied implication, set the implied (rightmost) variable to true. Return the truth assignments

DYNAMIC PROGRAMMING ALGS

longest increasing subsequence: $O(n^2)$: $L(j)$=len of longest subsequence ending @ j. $L(j) = 1 + \max\{L(i)\}$ where $i < j, a_i < a_j \rightarrow (i, j)$ is an edge. Return $\max\{L(i) for i \leq n\}$, and loop from $i = n \rightarrow 0$

edit distance: $O(m \cdot n)$: $E[m, n]$ =min num of edits to change $x[1 \cdots m]$ to $y[1 \cdots n]$. E[m,n] =

$$\min \begin{cases} \cdots x[m], \cdots - \\ \text{delete x[m]: } E[m,n] = 1 + E[m-1,n] \\ \cdots -, \cdots y[n] \\ \quad \text{insert y[n]: } E[m,n] = 1 + E[m,n-1] \\ \cdots x[m], \cdots y[n] \\ \quad \text{change x[m] to y[n]:} \\ \; E[m,n] = E[m-1,n-1] + \text{ (1 if diff, 0 else)} \end{cases}$$

Initialize $i = 0 \cdots m : E[i, 0] = i \quad j = 0 \cdots n : E[0, j] = j$. Loop $i = 1 \cdots m, j = 1 \cdots n, E[i, j]$

Knapsack: $O(W_{\max} \cdot n)$: $K(w)$ =max value of weight $w \leq W_{\max}$. $K(w) = \max_{w_i \leq w}\{v_i + K(w - w_i)\}$ $K(0) = 0$. Loop $w = 1 \cdots W_{\max}$

Chain Matrix Multiply: $O(n^3) : C(i, j)$ = cost of best solution to multiplying $A_i \cdots A_j$.

$$C(i,j) = \min_{i \leq k \leq j}\{C(i,k) + C(k+1,j) + m_{i-1} \cdot m_k \cdot m_j\}$$

Solve in order of increasing subproblem length: for $i = 1 \cdots n - 1 : C(i, i = 0)$. for $s = 1 \cdots n - 1$, for $i = 1 \cdots n - s : j = i + s$; update. Return $C(1, n)$.

Shortest path (all pairs of vertices): $O(|V| \cdot (|V| + |E|))$: $dist(v, i)$ = dist from s to v using i edges. $dist(v, i) = \min_{e=(w,v)}\{len(e) + dist(w, i - 1)\}$. For all $v \in V \quad dist(v, 0) = \infty, dist(s, 0) = 0$. for $i = 1 \cdot |V|$, for all $v \in V$, update.

Independent Sets: $O(|V| + |E|)$: $I(u)$ =size of largest independent set in subtree rooted at u.

$$I(u) = \max \begin{cases} 1 + \sum_{grandchildren} I(g_i) & \text{if } u \in I(n) \\ \sum_{children} I(c_i) & \text{u isn't} \end{cases}$$

DFS traversal: postvisit[u] = update (does u after children of u are done)

Travelling Salesman: $O(2^n n^2)$ Subset $S \leq V$, $\{1\} \in S$.

for $j \in S, C(S, j)$ =length of shortest path that starts at 1, ends at j, and visits each vertex in S once. If $|S| > 1, C(S, 1) = \infty$.

update: $C(S, j) = \min_{i \in S, i \neq j, (i,j) \in E}\{C(S - \{j\}, i) + len(i, j)\}$

For $S = 2 \cdots n$: for all sizes of S, for all subsets $S \leq \{1 \cdots n\}$ of size s (including $\{i\}$): $C(S, 1) = \infty$ for all $j \in S, j \neq 1, update$