longest increasing subsequence: $O(n^2)$: $L(j)$=len of longest subsequence ending @ j. $L(j) = 1 + \max\{L(i)\}$ where $i < j, a_i < a_j \to (i,j)$ is an edge. Return $\max\{L(i) for i \leq n\}$, and loop from $i = n \to 0$
edit distance: $O(m \cdot n)$: $E[m,n]$ =min num of edits to change $x[1 \cdots m]$ to $y[1 \cdots n]$.

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

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.

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

DFS traversal: postvisit[u] = update (does u after children of u are done)
Travelling Salesman: $O(2^n n^2)$