Reference Sheet

Unless explicitly stated otherwise, you may use any algorithm discussed in this class or 332 to solve a problem. In particular, you may use any of these functions as libraries (this list is not exhaustive).

Graphs

- TwoColor($G$) returns True if $G$ can be 2-colored (i.e., is bipartite), False otherwise. Running time $\Theta(m + n)$
- ConnectedComponents($G$) finds the connected components of an undirected graph $G$. You may assume you get any reasonable representation of this information. Running time $\Theta(m + n)$
- StronglyConnectedComponents($G$) finds the strongly connected components of a directed graph $G$. You may assume you get any reasonable representation of the information. Running time $\Theta(m + n)$
- TopologicalSort($G$) returns a list of vertices of a directed graph $G$ in topological order, or null if the graph has a cycle. Running time $\Theta(m + n)$
- CondensationGraph($G$) returns the condensation of a directed graph $G$ (a.k.a., the “meta-graph” of $G$ or “graph of SCCs of $G$). Running time $\Theta(m + n)$
- Prims($G$) finds the minimum spanning tree of a (weighted, undirected) graph $G$. Running time $\Theta(m \log n)$
- Dijkstra($G, s$) finds the length of the shortest path from $s$ to every vertex in a (non-negative) weighted, directed graph $G$. Running time $\Theta(m + n \log n)$. Finds the path itself for any target in $O(n)$ additional time.
- Bellman-Ford($G, s$) finds the length of the shortest path from $s$ to every vertex in a weighted, directed graph $G$. Detects negative weight cycles, if any. Running time $\Theta(mn)$. Can find the path itself for any target in $O(n)$ additional time.
- Floyd-Warshall($G$) finds the length of the shortest path between all pairs of vertices in a weighted, directed graph $G$. Detects negative weight cycles, if any. Running time $\Theta(n^3)$. Can find the path itself for any pair in $O(n)$ additional time.
- Ford-Fulkerson($G, s, t$) Finds a maximum flow from $s$ to $t$ and a minimum cut separating $s$ and $t$. Running time $\Theta(Ef)$, where $f$ is the value of the flow.

Arrays

- QuickSelect($A, k$) returns the value which would be at index $k$ of $A$ if $A$ were sorted. Running time $\Theta(n)$
- MaxSubarraySum($A$) returns the sum of the maximum sum (contiguous) subarray of $A$. Running time $\Theta(n)$
- MergeSort($A$) returns the sorted version of $A$. Running time $\Theta(n \log n)$

Others

- Gale-Shapley(riderPrefs, horsePrefs) returns the rider-optimal stable matching. Running time $\Theta(n^2)$ for $n$ riders and $n$ horses
- 2dClosestPoints($A$) returns the distance between the two closest points of $A$ (where $A$ contains vectors in $\mathbb{R}^2$). Running time $\Theta(n \log n)$
- EditDistance($x, y$) returns the edit distance between strings $x$ and $y$. Running time $\Theta(m + n)$ for strings of length $m, n$
- LP-Solver(vars, constraints, objective) returns the optimal feasible point for a linear program. Running time $\Theta(n^3)$ for an input written with $n$ bits.

There's more information on the back!
Other information

Master Theorem  For a recurrence of the following form, where $a, b, c, d$ are constants

$$T(n) = \begin{cases} 
  d & \text{if } n \text{ is at most some constant} \\
  aT\left(\frac{n}{b}\right) + f(n) & \text{otherwise} 
\end{cases}$$

Where $f(n)$ is $\Theta\left(n^c \cdot \log^k(n)\right)$ for $k \geq 0, a \in \mathbb{Z}^+, c \geq 1$

- If $\log_b(a) < c$ then $T(n) = \Theta\left(n^c \cdot \log^k(n)\right)$
- If $\log_b(a) = c$ then $T(n) = \Theta\left(n^c \cdot \log^{k+1}(n)\right)$
- If $\log_b(a) > c$ then $T(n) = \Theta\left(n^{\log_b(a)}\right)$

DFS Edge Classification  For directed graphs

<table>
<thead>
<tr>
<th>Edge type</th>
<th>Definition</th>
<th>$(u, v)$ is of this type if and only if</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>Edges forming the DFS tree</td>
<td>$v$ was not seen before we processed $(u, v)$</td>
</tr>
<tr>
<td>Forward</td>
<td>From ancestor to descendant in tree</td>
<td>$u, v$ both seen; $u.\text{start} &lt; v.\text{start} &lt; v.\text{end} &lt; u.\text{end}$</td>
</tr>
<tr>
<td>Back</td>
<td>From descendant to ancestor in tree</td>
<td>$u, v$ both seen; $v.\text{start} &lt; u.\text{start} &lt; u.\text{end} &lt; v.\text{end}$</td>
</tr>
<tr>
<td>Cross</td>
<td>$u, v$ have no ancestor/descendant relationship</td>
<td>$u, v$ both seen; $u.\text{start} &lt; v.\text{end} &lt; u.\text{start} &lt; u.\text{end}$</td>
</tr>
</tbody>
</table>

NP-Complete Problems  The following problems are NP-complete

- **$k$-COLOR:** Given a graph $G = (V, E)$ and an integer $k$ (where $k \geq 3$), return true if there is a function $f : V \rightarrow \{1, \ldots, k\}$ such that if $(u, v) \in E$ then $f(u) \neq f(v)$.

- **VERTEX-COVER:** Given a graph $G = (V, E)$ and an integer $k$, return true if there is a set of vertices $S$, such that $|S| \leq k$ and $\forall (u, v) \in E : u \in S \lor v \in S$.

- **CLIQUE:** Given a graph $G = (V, E)$ and an integer $k$, return true if there is a set of vertices $S$, such that $|S| \geq k$ and $\forall u, v : [u \neq v \land u, v \in S] \rightarrow (u, v) \in E$.

- **IND-SET:** Given a graph $G = (V, E)$ and an integer $k$, return true if there is a set of vertices $S$, such that $|S| \geq k$ and $\forall u, v : [u \neq v \land u, v \in S] \rightarrow (u, v) \notin E$.

- **3-SAT:** Given an expression in CNF form, where each clause contains exactly three literals, return true if there is a setting of the variables that causes the expression to evaluate to true.

- **HAM-PATH:** Given a directed graph $G$, return true if there is a Hamiltonian Path in $G$, that is a path that visits each vertex exactly once.