# DP and Graph Algorithms
# Problem Set 2 – CS 6515/4540 (Fall 2025)

This problem set is due on **Thursday, September 11th**. Submission is via Gradescope. Your solution must be a typed pdf (e.g. via LaTeX) – no handwritten solutions.

## 4 DP: Balancing Array

Given an array of $n$ **nonnegative** integers $A$ and an integer $k$, we want to partition $A$ into $k$ contiguous nonempty subarrays such that the subarray with the largest sum is minimized. Let the minimum achievable largest sum be $S$. Provide an algorithm to compute $S$ as efficiently as possible using Dynamic Programming.

*Example:* $A = [1, 4, 6, 2, 5], k = 3$. In this case, the most balanced subarray partition of $A$ is $[1, 4], [6], [2, 5]$ and we have $S = \max(1 + 4, 6, 2 + 5) = 7$.

1. Define the entries of your table in words. E.g. $T(i)$ or $T(i, j)$ is ...
2. State recurrence for entries of the table in terms of smaller subproblems. Briefly explain in words why it is correct.
3. Analyze the running time of your algorithm. It should be $O(kn \log n)$

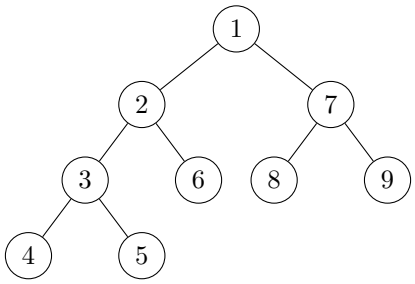
**Remark:** There is a more efficient way to solve this without using DP that eliminates any dependency on $k$.

## 5 DP: Minimum Lights

The Georgia Tech campus spans over $n$ academic buildings, with pathways that interconnect resembling a tree $T = (V, E)$. The campus police want to ensure all pathways are illuminated at night. Lights can be installed on any building $B \in V$, and when done so, all pathways/edges connected to $B$ are lighted.

Your goal is to develop a strategy using Dynamic Programming to find the minimum number of lights needed to ensure the entire campus is well-lit at night. For this, you are given an unweighted undirected tree structure with $n$ vertices and $m$ edges, where each node represents an academic building and the edges represent pathways. Find some $S \subseteq V$ such that every edge $e \in E$ is incident to at least one vertex in $S$, and $|S|$ is minimized. The running time of your algorithm should be $\mathcal{O}(|V|)$.

*Example Input*:



*Example Output*: 3. The buildings that need to lit up are {2, 3, 7}.

1. Define the entries of your table in words. E.g. $T(i)$ or $T(i, j)$ is ...
2. State recurrence for entries of the table in terms of smaller subproblems. Briefly explain in words why it is correct.
3. Analyze the running time of your algorithm.

# 6 Knapsack and MDP

## 6.1 Knapsack

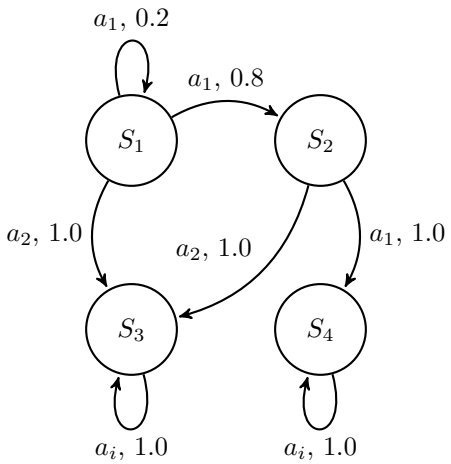
Recall the Knapsack problem where we have budget $B$ and $n$ items labeled $i$ where $1 \leq i \leq n$, and item $i$ has weight $s_i \leq B$ and valuation $v_i$, and we want to compute the subset with the maximum total valuation $\sum_{i \in I} v_i$ of all subsets $I \subseteq [n]$ such that $\sum_{i \in I} s_i \leq B$

1. Show that the PTAS given in lecture where we apply the $O(n^2 \max v_i)$ DP algorithm but with valuations truncated to their nearest smaller multiple of $K$, will give a feasible subset with total valuation $V$, which satisfies $1 - \frac{nK}{\max v_i} \leq \frac{V}{V'} \leq 1$, where $V'$ is the optimal valuation of the original problem.
2. Choose the appropriate $K$ in terms of $\epsilon$ and $\max v_i$ such that we get $\frac{V}{V'} \geq 1 - \epsilon$ in poly$(n, \frac{1}{\epsilon})$ time.
3. Suppose there exists an approximation algorithm that could, in polynomial (NOT pseudo polynomial) time, get a $1 - \epsilon$ approximation to Knapsack for $\epsilon < \frac{1}{n \max v_i}$. Then, show that $P = NP$.

## 6.2 MDP

Consider the MDP with 4 states $\{S_1, S_2, S_3, S_4\}$ and 2 actions $\{a_1, a_2\}$ as given in the figure. Here, the tuple $a, p$ on an edge means that if we perform action $a \in \{a_1, a_2\}$ at the start state then we end at the destination state with probability $p$. (When we write $a_i$ in the figure that means this is true for both $a_1$ and $a_2$.) Moreover, we receive reward 10 on pulling any action from $S_3$, we receive reward 100 on pulling any action from $S_4$, and we receive reward 0 on pulling any action from $S_1$ or $S_2$.

Suppose we start at $S_1$, then what is the optimal policy for time horizon $T = 2$ (i.e., when you are only allowed 2 pulls)? Justify your answer in a few words.



# 7 Walking on the Plane

Consider $n$ points on the Euclidean plane. Each point is the center of a dynamically changing circle whose radius grows linearly with time, i.e. the radius of each circle at time $t$ is $t$.

Given two vertices $u$ and $v$, give an algorithm that finds the earliest time $t$ at which $u$ and $v$ become connected - i.e. you can freeze time and walk from $u$ to $v$ on the Euclidean plane and be inside at least one circle at any point in your path. Provide the time complexity and justify the correctness of your algorithm. It must run in $\widetilde{O}(n^2)$ time [1]

Your solution can only utilize algorithms taught in lecture.

**Remark:** This problem can be solved in $O(n \log n)$ time by computing the Delaunay Triangulation.

# 8 Bipartite Matching: Forbidden Paths

We are given a directed graph $G = (V, E)$ with $V = \{v_1, ..., v_n\}$ and a forbidden set $F = \{f_1, f_2, \ldots, f_k\} \subset V$.

Define a bipartite graph $\widehat{G} = (L \cup R, \widehat{E})$ as follows: $L = \{v_1, ..., v_n\}$, $R = \{v'_1, ..., v'_n\}$ (so we have two copies of $V$, one on the left and one on the right side of the bipartite graph). Edge $\{v_i, v'_i\} \in \widehat{E}$ for all $i = 1, ..., n$, and $\{v_i, v'_j\} \in \widehat{E}$ for all $(v_i, v_j) \in E$. Given two vertices $v_s, v_t \in V$, delete $v'_s$ from $R$ and delete $v_t$ from $L$. We also delete $f_i$ from $L$ and $R$ for each $f_i \in F$ (when we delete these two vertices, we must also delete any edges from $\widehat{E}$ that $v'_s$ and $v_t$ were connected to).

**Problem:** Show that a perfect matching exists in $\widehat{G}$ if and only if there exists a directed path from $v_s$ to $v_t$ in $G$ that avoids all vertices in F.

[1] $\widetilde{O}$ means we don't care about sub-polynomial factors like $\log n$. For example, $n^2 \log^2(n) = \widetilde{O}(n^2)$ but $n^{2.3}$ is not.