CSE 3500 Practice DP/D&C Questions for Exam 1

Here are some problems on dynamic programming and divide and conquer algorithms. A problem similar to at least one of these will be on the test.

Finding the face dancers

There has been a recent infiltration of face dancers on UConn's campus. Face dancers are shapeshifters with the ability to mimic other individuals, making them popular for numerous roles, including spies and assassins. The teaching staff of CSE 3500 have been able to narrow down the suspects across the UConn Storrs campus (red X's in Figure 1).

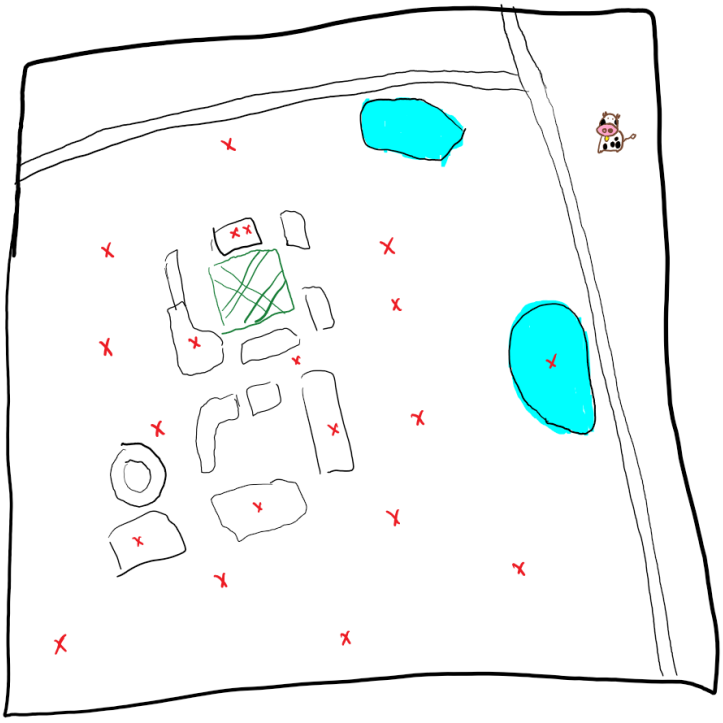

Figure 1: Face dancer locations in Storrs.

There are exactly 2 face dancers on campus and the teaching staff know that they are very close to each other. **Problem:** Given n locations defined as $(x_1, y_1), \ldots, (x_n, y_n)$ coordinates on the UConn Storrs campus, your task is to design an algorithm that finds the pair of points that are closest together. Closeness is defined as Euclidean distance $d((x_i, y_i), (x_j, y_j)) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. *Hint: First think about how you might do this in one dimension. In two dimensions, think about splitting the map in Figure 1 up.*

CSE 3500 **Practice DP/D&C Questions for Exam 1**

Quicksort

Quicksort is a fast divide-and-conquer sorting algorithm that works in a similar way to the Selection algorithm we learned in class. Consider the quicksort algorithm in Figure 2.

```
Quicksort(S):
If |S| ≤ 3 then
   Sort S
   Output the sorted list
Else
   Choose a splitter a_i ∈ S uniformly at random
   For each element a_j of S
      Put a_j in S^- if a_j < a_i
      Put a_j in S^+ if a_j > a_i
   Endfor
   Recursively call Quicksort(S^-) and Quicksort(S^+)
   Output the sorted set S^-, then a_i, then the sorted set S^+
Endif
```

Figure 2: Quicksort algorithm.

1. What is the worst case runtime of this Quicksort algorithm?
2. How can we modify the Quicksort algorithm to make our average runtime analysis easier? (Hint: we can modify it to behave more similarly to the Selection algorithm)
3. What is the Quicksort loop invariatn? Prove that this algorithm is correct by showing three properties of the Quicksort invariant holds. That is
   (a) Initialization: The invariant is true before the first iteration.
   (b) Maintenance: If the invariant is true before an iteration, it is true after an iteration.
   (c) Termination: When the code (e.g. a loop) terminates, the invariant gives a useful property that helps prove the algorithm is correct.

Dynamic Programming

Having recently watched Tarantino's Reservoir Dogs, you decide to become a diamond thief. In preparation for your upcoming heist, you think about which algorithm to follow to max-

imize your diamond haul. Given that there are n diamonds, each with weight w_i where $i = 1, \ldots, n$ and value v_i where $i = 1, \ldots, n$, your goal is to select the subset S of items such that $\sum_{i \in S} w_i \leq W$ where W is the weight of your backpack and $\sum_{i \in S} v_i$ is maximum.

- What is the intermediate subproblem that we can use to build solutions to larger subproblems?
- How big is the dynamic programming array?
- Give the recurrence relation for this problem.

Heapy question

1. Show the steps to convert the min heap in Figure 3 to a max heap using the max-heapify algorithm.
2. Give a tight asymptotic bound on the number of nodes in a heap and binary search tree in terms of its height h and the number of nodes n.

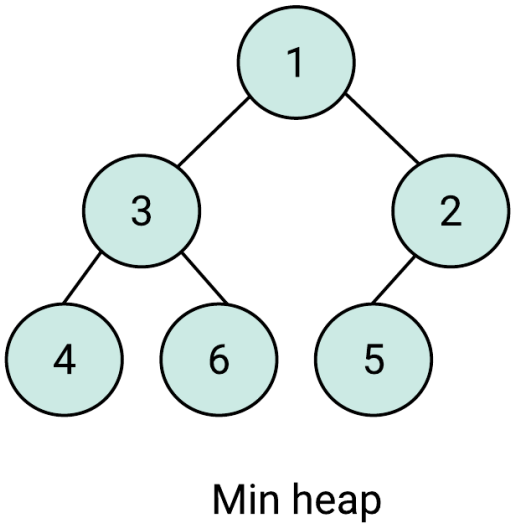

Figure 3: Min heap.