• Don’t forget to put your name and ID on the cover page
• This exam is closed-book
• If you have a question, stay seated and raise your hand.
• Please try to write legibly – if I cannot read it, you may not get credit.
• Do not waste time – if you cannot solve a question immediately, skip it and return to it later.
• Try your best to answer each question. Partial credits will be given if you show that you have some ideas – but not according to the length of your answer.
• Be succinct.

1) Sorting and Selection 20
2) Quicksort 5
3) Heaps 23
4) Hash Tables 7
5) Longest Common Subsequence 15
6) Restaurant Location Problem 15
7) Shortest Paths 15
8) (Extra credit) Algorithm Design 20
Total 120
1. (20 points) Sorting and Selection

Suppose that you have an array of \( n \) integers in the range \([1, k]\), and you would like to find the median of the array. You are mainly considering two choices: (1) using an order-statistics algorithm, such as the randomized selection algorithm or the worst-case linear-time selection algorithm, or (2) using an sorting algorithm to sort the array and then choose the median from the sorted array.

a. Fill in the table below with the time and memory complexities (in \( \Theta \)) of the algorithms that you can choose from. To determine the (extra) memory complexity, exclude the space for the input array.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Expected time complexity</th>
<th>Worst-case time complexity</th>
<th>Extra memory complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomized Quicksort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merge Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counting Sort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Randomized Selection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear-time Selection</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. When \( k \in \Theta(n) \), which algorithm(s) would you prefer and why?

c. When \( k \) is much larger than \( n \), which algorithm(s) would you prefer and why?

2. (5 points) Quicksort.

Show how quick sort can be made to run in \( \Theta(n \log n) \) time in the worst case, assuming all elements are distinct. (Hint: you need a good strategy to select the pivot.)
3. (23 points) Heaps

Note: by default, we mean a max-heap, i.e., the largest element is at the root.

a. Is the tree below a heap? Why or why not?

b. Illustrate how buildheap works using the following example. Each step is the result of one call to the heapify function.
c. Starting from the heaps on the left, show the contents of the new heaps after each heap operation.

![Heap Diagrams]

- Insert 14
- ExtractMax
- Decrease the 2nd key (12) to 5

d. What is the time complexity to build a min heap with $n$ elements?

e. The procedure BuildHeap can also be implemented by repeatedly using HeapInsert to insert the elements into the heap. Consider the following implementation:

```java
BuildHeap(A)
    heapsize(A) = 1;
    for (i = 2 to length(A))
        HeapInsert(A, A[i]);
```

What is the worst-case time complexity of above procedure? Briefly justify your answer.
4. (7 points) Hash tables.

Demonstrate the insertion of the keys 14, 2, 8, 19, 18, 12, 9 into a hash table with collision resolved by chaining. Let the table have \( m = 7 \) slots, and let the hash function be \( h(k) = k \mod m \).

5. (15 points) Longest common subsequence (LCS).

a. Complete the following dynamic programming table to compute the length of LCS between two strings ACDBADAC and ACBADCAB.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>x[i]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. What is the actual longest common subsequence? If there are multiple longest common subsequences, report all of them.
6. (15 points) Restaurant location problem.
Solve the following optimal restaurant location problem using dynamic programming. The
distance constraint is that two selected locations cannot be within 10 miles.

<table>
<thead>
<tr>
<th>Distance between locations (mi)</th>
<th>5  2  4  6  6  3  6  9  7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Profit ($10k)</td>
<td>15  17  4  11  9  2  8  5  6  8</td>
</tr>
</tbody>
</table>

Total Profit
Select $i$
Do not select $i$
Best

7. (15 points) Shortest path problem.
Use dynamic programming to find the shortest path from node S to node G on the following
directed graph. Note that one can only move following the directions of the arrows. For full credit,
show both the shortest path and its length.
8. (20 points) Extra credit: Algorithm design.

a. (10 points) Let $X[1..n]$ and $Y[1..n]$ be two arrays, each containing $n$ numbers already in sorted order. Describe an $\Theta(\log n)$-time algorithm to find the median of the $2n$ elements in arrays $X$ and $Y$. Be succinct.

b. (10 points) A palindrome is a nonempty string over some alphabet that reads the same forward and backward. Design an efficient algorithm that takes a sequence $x[1\ldots n]$ and returns the longest palindromic subsequence. What is the running time of your algorithm?

(For instance, the sequence ACGTGTCAAAAATCG has many palindromic subsequences, including ACGCA and CAAC. On the other hand, the subsequence ACG is not palindromic.)
Blank page
Blank page
Blank page