Topics that might be on the second examination on October 26 2022

A few time complexity and recurrence problems, since not everyone has mastered them.

Be able to write pseudocode:
1. Floyd Warshall algorithm. W are the arc weights.

For all i and all j V[i,j] = W[i,j], back[j] = i

For all j
For all i
For all k
{ temp = V[i,j]+W[j,k]
  if(temp < V[i,k])
    V[i,k] = temp
back[i,k] = back[j,k]
}

2. Bellman Ford algorithm. 0 = source.

For all i V[i] = W[0,i], back[i] = 0
Repeat the following until there are no more changes
{ for all arcs (i,j)
  { temp = V[i]+W[i,j]
    if(temp < V[j])
      V[j] = temp
back[j] = i
  }
}

Be able to step through Dijkstra’s algorithm for a small graph.

Johnson’s algorithm.
Given a figure showing a weighted directed graph with some negative weights, Update the weights on edges so that there are no negative weights.

What is the worst-case time complexity for each of those four algorithms?

Johnson’s O(nm logn) Floyd Warshall Theta(n^3) Bellman Ford O(nm) Dijkstra O(m logn)

Dynamic Programming, possibly with memoization.
Example problem: maximum sum contiguous subsequence. Given a sequence of numbers, each either positive or negative, find a contiguous subsequence of maximum total. As an example if the input sequence is 3,-4,8,-2,3,5,-1,6,-5 the output sequence is 8,-2,3,5,-1,6. There are several algorithms for this problem.

1. Dumb Brute Force: $O(n^3)$ time.
2. Slightly Smarter Brute Force: $O(n^2)$ time.
3. Divide and Conquer: $O(\log n)$ time.
4. Dynamic Programming: $O(n)$ time.

BFPRT (median of medians) algorithm for Selection.

Explain how to obtain the recurrence for the time complexity, that is $T(n) \leq T(n/5)+T(7n/10)+n$.

It takes $O(1)$ time to find the median of all blocks of 5. That's $\Theta(n)$. Finding the median of those medians, to use as a pivot, takes $T(n/5)$ time. Using the pivot, we partition the set into two pieces. Each piece has at least $3n/10$ items, hence each piece has no more than $7n/10$ items.

Find components of an undirected graph using union/find.

Here is an example. The graph is given to us as a list of edges. Let the vertices be Roman letters in the range A .. J. Let the list of edges be:

Work this example.

Each time you see an edge, you do find for both vertices, then union for both leaders if they are different.

The final diagram is shown below.

Heapsort. $O(n \log n)$ time. A sped up version of selection sort.