B1 Algorithms (25 points)

Do both parts.

a) [15 points] For each of the three algorithms below give a brief explanation of how the algorithm works. You should capture the main idea/ideas of the algorithm and state the running time.

i) FFT

ii) Dijkstra's

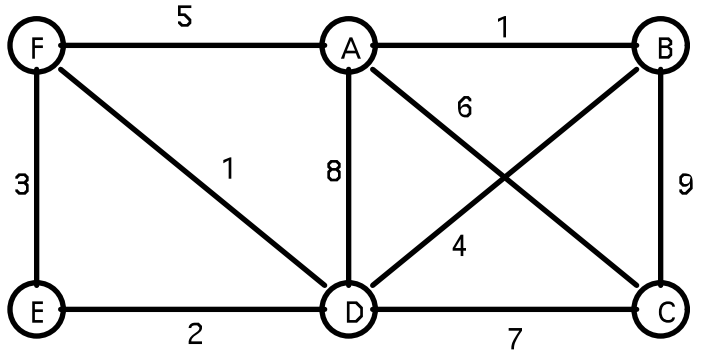
iii) Strassen's

b) An early (circa 1926) algorithm for finding a minimum spanning tree in a weighted connected undirected graph G, with vertices V, edges E, and edge weights w, is Boruvka's algorithm:

```
Input: undirected graph G = (V, E, w)
Output: T, a minimum spanning tree of G

Let T be the graph consisting of the vertices in V with no edges;
while T has more than one connected component do
   for each connected component C of T do
       Let e be a min cost edge (v,u) such that v is in C and u is not in C;
       T = T ∪ {e};
```

i) [3 points] When Boruvka's algorithm is applied to the following graph, state for each iteration of the while-loop the edges that are added to T in that iteration:



ii) [7 points] Carefully explain how to implement Boruvka's algorithm in time O(e log v).

B2 Algorithms (25 points)

Do all three parts.

a) [5 points] Formally define the following terms (you can assume that P and NP are known terms):

i) polynomially reducible

ii) NP-complete

b) [9 points] There is often a thin line between a problem being NP-complete and being solvable in polynomial time. We know for instance that the problems Graph Coloring, Hamiltonian Circuit and Clique are all NP-complete. Nonetheless, for each of the following three problems, briefly explain how to solve it in polynomial time:

i) Determining if an undirected graph is 2-colorable.

ii) Determining if an undirected graph contains an Eulerian circuit.

iii) Determining if an undirected graph contains a clique of size 4.

c) [11 points] The problem known as **Hitting Set** is the following:

INSTANCE: A set W, subsets $C_1, C_2, \ldots, C_n$ of W, and an integer k.
Note that the subsets $C_1, C_2, \ldots, C_n$ may be overlapping.

QUESTION: Does there exist a set H, where $|H| \leq k$ such that H contains at least one element from each of $C_1, C_2, \ldots, C_n$? Such a set H is a Hitting Set.

Give a full proof that Hitting Set is NP-complete. You may only assume that the following three problems are known to be NP-complete:

Partition
INSTANCE: n numbers
QUESTION: Can the n numbers be partitioned into two sets so that the numbers in each set sum to the same value?

Vertex Cover
INSTANCE: Undirected graph G, integer k
QUESTION: Does G contain a vertex cover of size no more than k?

Graph Coloring
INSTANCE: Undirected graph G, integer k
QUESTION: Is there a k-coloring of the vertices of G?

B3. Algorithms (25 points)

Do all three parts

a) [12 points] Mergesort.

   i) Describe how the Mergesort algorithm works and state its worst-case running time.

   ii) How many key comparisons are done by Mergesort if the keys are already in order when the sort begins?

   iii) What is a stable sorting method? Is Mergesort a stable sorting method? Justify your answer.

b) [7 points] Suppose E1 and E2 are arrays, each with n keys sorted in ascending order. Devise an O(log n) algorithm to find the n-th smallest of the 2n keys, namely the median of the combined set. For simplicity, you can assume the keys are distinct.

c) [6 points] Let E be an array containing n integers. Design an algorithm that runs in linear time, i.e., O(n), to find the maximum sum for a contiguous subsequence of elements of E. If all elements of E are negative, the maximum contiguous subsequence is defined to be the empty sequence with sum equal to zero. For example, consider E to be the following sequence of elements

   12, -15, 18, -3, 14, -15, -6, 8, 7, -5, 2

   The maximum subsequence sum (MSS) is 29, starting at position 3 and ending at position 5. Your algorithm does not need to specify the starting and ending position of MSS.

B4 Algorithms (25 points)

Do all four parts.

a) [6 points] Heaps

i) Describe the structure of a Heap. Be sure to include all items that are associated with each node. Note that you should just describe the structure – do not describe the associated operations or how they are implemented.

ii) Heaps typically implement just the standard priority queue operations of Min, Insert and Delete_min. However it is possible to implement a general Delete operation in a heap under the assumption that the operation is provided with a pointer to the object in the heap that is to be deleted. Explain how, under that assumption, you could implement a general Delete in time O(log n).

b) [10 points] Fibonacci Heaps

i) Describe the structure of a Fibonacci Heap. Be sure to include all items that are associated with each node. Note that you should just describe the structure – do not describe the associated operations or how they are implemented.

ii) What is the worst-case (non-amortized) running time of a Delete_min operation in a Fibonacci Heap? Explain your answer.

iii) The most commonly used potential function for Fibonacci Heaps is $t + 2m$, where t is the number of trees in the root list and m is the number of marked nodes. Using that potential function, derive (show your calculations) the amortized running time of these operations:

- Delete_min
- Decrease_key

c) [6 points] Dictionary

i) Name two dictionary data structures that use O(log n) worst case time for each of the three standard dictionary operations. Note: Just name those data structures – you do not need to describe them.

ii) Name one dictionary data structure that uses O(1) average case time for each of the three standard dictionary operations. Note: Just name that data structure – you do not need to describe it.

d) [3 points] Real time scheduling systems often maintain a set of *pending tasks*. These pending tasks are waiting to be executed and, associated with each task is the time at which it is to be executed. Additional pending tasks can join the system at any point and can have any execution time equal to or greater than the actual time. The next task to be executed is one with the earliest execution time among the pending tasks. What data structure should be used to maintain the set of pending tasks? Explain your answer.