Problem 1

now exploring edge(3,5)

the parent is [1, 2, 3, 4, 3, 6, 7, 8, 9]

MST now is [(3, 5)]

union structure {1, 2, 3(5), 4, 6, 7, 8, 9}

----

now exploring edge(4,6)

the parent is [1, 2, 3, 4, 3, 4, 7, 8, 9]

MST now is [(3, 5), (4, 6)]

union structure {1, 2, 3(5), 4(6), 7, 8, 9}

----

now exploring edge(3,4)

the parent is [1, 2, 3, 3, 3, 4, 7, 8, 9]

MST now is [(3, 5), (4, 6), (3, 4)]

union structure {1, 2, 3(4(6), 5), 7, 8, 9}

----

now exploring edge(2,9)

the parent is [1, 2, 3, 3, 3, 4, 7, 8, 2]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9)]

union structure {1, 2(9), 3(4(6), 5), 7, 8}

----

now exploring edge(4,5)

the parent is [1, 2, 3, 3, 3, 4, 7, 8, 2]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9)]

union structure {1, 2(9), 3(4(6), 5), 7, 8}

----

now exploring edge(6,9)

the parent is [1, 3, 3, 3, 3, 3, 7, 8, 2]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9)]

union structure {1, 3(2(9), 4(6), 5), 7, 8}

----

now exploring edge(5,9)

the parent is [1, 3, 3, 3, 3, 3, 7, 8, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9)]

union structure {1, 3(2(9), 4(6), 5), 7, 8}

----

now exploring edge(1,6)

the parent is [3, 3, 3, 3, 3, 3, 7, 8, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6)]

union structure {3(1, 2(9), 4(6), 5), 7, 8}

----

now exploring edge(4,8)

the parent is [3, 3, 3, 3, 3, 3, 7, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8)]

union structure {3(1, 2(9), 4(6), 5, 8), 7}

----

now exploring edge(1,5)

the parent is [3, 3, 3, 3, 3, 3, 7, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8)]

union structure {3(1, 2(9), 4(6), 5, 8), 7}

----

now exploring edge(1,8)

the parent is [3, 3, 3, 3, 3, 3, 7, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8)]

union structure {3(1, 2(9), 4(6), 5, 8), 7}

----

now exploring edge(7,8)

the parent is [3, 3, 3, 3, 3, 3, 3, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8), (7, 8)]

union structure {3(1, 2(9), 4(6), 5, 7, 8)}

----

now exploring edge(5,7)

the parent is [3, 3, 3, 3, 3, 3, 3, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8), (7, 8)]

union structure {3(1, 2(9), 4(6), 5, 7, 8)}

----

now exploring edge(5,6)

the parent is [3, 3, 3, 3, 3, 3, 3, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8), (7, 8)]

union structure {3(1, 2(9), 4(6), 5, 7, 8)}

----

now exploring edge(6,7)

the parent is [3, 3, 3, 3, 3, 3, 3, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8), (7, 8)]

union structure {3(1, 2(9), 4(6), 5, 7, 8)}

----

now exploring edge(3,7)

the parent is [3, 3, 3, 3, 3, 3, 3, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8), (7, 8)]

union structure {3(1, 2(9), 4(6), 5, 7, 8)}

----

now exploring edge(2,4)

the parent is [3, 3, 3, 3, 3, 3, 3, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8), (7, 8)]

union structure {3(1, 2(9), 4(6), 5, 7, 8)}

----

now exploring edge(2,3)

the parent is [3, 3, 3, 3, 3, 3, 3, 3, 3]

MST now is [(3, 5), (4, 6), (3, 4), (2, 9), (6, 9), (1, 6), (4, 8), (7, 8)]

union structure {3(1, 2(9), 4(6), 5, 7, 8)}

Problem 2.

a. Use Inductive method.

<u>Inductive Hypothesis</u>: Assume G be a weighted undirected graph (V, E), T be a MST of G, if e is an edge in T, then $T-\{e\}$ has exactly two connected components.

<u>Base</u>: If there is one edge e in ~~T~~, $T-\{e\}$ will be empty, i.e., exist two individual nodes, absolutely two connected part.

<u>Maintenance</u>:

hypothesis: if e is an edge in T, $|T|=K$, $T-\{e\}$ has two connected compnonents.

Let's prove, $|T|=K+1$, $T-\{e\}$ has two connected components.

As the "cut" defined in textbook, cut on T, thus $T=\{X\}+\{T-X\}$ (X is a node of T). According to textbook, $\{X\}, \{T-X\}$ are two connected components separetly. ⟨1⟩ if $e=$ ~~X~~, thus $\{e\}, \{T-e\}$ are two separately connected components ⟨2⟩ if $e \neq X$, according to our hypothesis, $\{T-e\}$ and $\{e\}$ are two connected components, however, X must connects to one of $\{T-e\}$, $\{e\}$, thus there are still two connected components.

b. ~~Assume T_1, T_2 are two connected components of $T-\{e\}$. (according to cut in textbook). N_1, N_2 are the node set of T_1, T_2.~~
~~P 136~~.

Assume C and C' are two connected components of $T-\{e\}$. (according to what proved in a, there only exist two part).

Because e' is an edge that crosses the cut C, which means the other one vertex of e' must be in C', for there are only two connected components ~~of $T-\{e\}$~~ exists (conclusion a.). Besides, e' and e has the same weight, so $T-\{e\} \cup \{e'\}$ is also a MST.

Problem 3

---step 1---

cost is

(1 : inf) (2 : inf) (3 : inf) (4 : inf) (5 : 0.0) (6 : inf) (7 : inf) (8 : inf) (9 : inf)

Heap is

5.cost= 0.0

1.cost= inf

3.cost= inf

4.cost= inf

2.cost= inf

6.cost= inf

7.cost= inf

8.cost= inf

9.cost= inf

1.visited = False

2.visited = False

3.visited = False

4.visited = False

5.visited = True

6.visited = False

7.visited = False

8.visited = False

9.visited = False

---step 2---

cost is

(1 : inf) (2 : 8.0) (3 : inf) (4 : 5.0) (5 : 0.0) (6 : 9.0) (7 : inf) (8 : 5.0) (9 : inf)

Heap is

4.cost= 5.0

8.cost= 5.0

6.cost= 9.0

2.cost= 8.0

9.cost= inf

3.cost= inf

7.cost= inf

1.cost= inf

1.visited = False

2.visited = False

3.visited = False

4.visited = True

5.visited = True

6.visited = False

7.visited = False

8.visited = False

9.visited = False

---step 3---

cost is

(1 : 7.0) (2 : 4.0) (3 : 4.0) (4 : 5.0) (5 : 0.0) (6 : 9.0) (7 : 9.0) (8 : 5.0) (9 : 2.0)

Heap is

9.cost= 2.0

2.cost= 4.0

3.cost= 4.0

1.cost= 7.0

8.cost= 5.0

6.cost= 9.0

7.cost= 9.0

1.visited = False

2.visited = False

3.visited = False

4.visited = True

5.visited = True

6.visited = False

7.visited = False

8.visited = False

9.visited = True

Problem 4.

General Idea: Sort the tasks according to their required time.
And the order should be the sorted order (increasingly).
Considering the time complexity, apply quick-Sort to sort.

<u>Algorithm</u>:

Input: tasks with require time of each task

Output: the order that minimize the total time until each task finished.

return quick-sort (tasks, required time).

For it is a quick-sort, time complexity is $O(n\log n)$.

Algorithm analysis: our order $W_1 W_2 \cdots W_n$ is the optimal order.

Let ~~as~~ then prove.

Assume W_i, W_j are two random tasks in our order, where $1 \le i < j < n$.

Because we have sorted tasks in increasing order, so $t_i \le t_j$.

Let swap W_j, W_j to make a new order.

The total time $T_{original} = t_1 + (t_1+t_2) + \cdots + (t_1+t_2+\cdots t_i) + \cdots + (t_1+t_2+\cdots+t_j)$
$+ (t_1+t_2+\cdots t_n)$

The new time $T_{new} = t_1 + (t_1+t_2) + \cdots + (t_1+t_2+\cdots+t_j) + \cdots + (t_1+t_2+\cdots+t_i)$
$+ (t_1+t_2+\cdots+t_n)$

$$T_{new} - T_{original} = (j-i)(t_j - t_i) \ge 0.$$

Thus. $T_{new} \ge T_{original}$. We prove that one change of two tasks will always make the $T_{new} \ge T_{original}$. Thus no matter how many tasks will change. T_{new} will always greater than or equal to $T_{original}$. So the original order is the opitimal one.

Problem 5

Letter	Frequency	Encode
A	20	00
B	29	10
C	10	010
D	2	111110
E	3	111111
F	4	11110
G	9	1110
H	11	011
I	12	110

Total bits:

20*2 + 29*2 + 10*3 + 2*6 + 3*6 + 4*5 + 9*4 + 11*3 + 12*3 = 283

The result is :

(('A', ('C', 'H')), ('B', ('I', ('G', ('F', ('D', 'E'))))))