***Kostaq Papa***

***0412768***

***CS2134***

***HW06***

**Written Part:**

1. Determine the running time of the function ***shortestpaths*** in the code in *bfs.cpp*

when |V | is the number of vertices and |E| is the number of edges. How does

the running time of this function change if *map<string, list<string> >* is used

instead of *vector<list<int> >* to store the graph? Determine the running time

of the function if this change is made.

Because the running time of the implementation of a queue is based on the number of edges, the running time of the shortestpashts function is O(|E|+|V|). If there were used maps instead of vectors, it would take longer to run the shortestpaths function. Because in the edges of the vertex, which is implemented with a queue, can be an empty string, the map.find() method will be needed each time to access the information. For the vector the operator [] takes O(1) time, and the find method takes O(log|V|) time. Therefore I can assume the new running time might be O(|V|+ log|V|+|E|).

1. Consider the following array representation of a binary heap:

[sentinel, 5, 29, 14, 88, 40, 77, 70, 100, 999, 45]

1. Show the tree representation of the binary heap.

5

70

100

459

77

14

88

29

40

999

1. Insert 22 into the binary heap; show both the tree representation and the array representation after 22 has been inserted.

Binary Heap: [sentinel, 5, 22, 14, 88, 29, 77, 70, 100, 999, 45, 40]

70

77

14

5

40

999

459

88

100

29

22

1. (a) Show the array representation

Binary Heap: [sentinel, 5, 29, 55, 88, 40, 89, 59, 100, 999, 45]

b) Show what happens when the root is removed by giving the tree representation

of the binary heap.

999

88

100

59

89

45

55

40

29

1. For the graph in the \_le Homework 6 Graphs, show the adjacency list representation

of the graph:

Shanghai :-> Tokyo -> Sydney

Tokyo: -> Los Angeles -> Hong Kong -> Sydney

Los Angeles: -> Tokyo -> Sydney

Hong Kong: -> Shanghai -> Sydney

Sydney: -> Hong Kong -> Los Angeles -> Auckland

Auckland: -> Sydney -> Rarotonga

Rarotonga: -> Auckland

Shortest path is: Shanghai-> (1) Sydney-> (2) Auckland-> (3) Rarotonga

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Distance/Predecessor** | | | | | | | **Visiting** | **Queue** |
|  | Sh | T | L | H | Sy | A | R |  |  |
| Init | 0/- |  |  |  |  |  |  |  | Sh |
| 1 | 0/- | 1/Sh |  |  | 1/Sh |  |  | Sh | T Sy |
| 2 | 0/- | 1/Sh | 2/T | 2/T | 1/Sh |  |  | T | Sy L H |
| 3 | 0/- | 1/Sh | 2/Sy | 2/Sy | 1/Sh | 2/Sy |  | Sy | L H A |
| 4 | 0/- | 1/Sh | 2/Sy | 2/Sy | 1/Sh | 2/Sy |  | L | H A |
| 5 | 0/- | 1/Sh | 2/Sy | 2/Sy | 1/Sh | 2/Sy |  | H | A |
| 6 | 0/- | 1/Sh | 2/Sy | 2/Sy | 1/Sh | 2/Sy | 3/A | A | R |
| 7 | 0/- | 1/Sh | 2/Sy | 2/Sy | 1/Sh | 2/Sy | 3/A | R |  |

1. Shortest weighted path is: Shanghai->(4)Tokyo->(14) Sydney->(16) Auckland->(20) Rarotonga

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Distance/Predecessor** | | | | | | | **Visiting** | **Queue** |
|  | Sh | T | L | H | Sy | A | R |  |  |
| Init | 0/- | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |  | Sh |
| 1 | 0/- | 4/Sh | ∞ | ∞ | 16/Sh | ∞ | ∞ | Sh | T Sy |
| 2 | 0/- | 4/Sh | 14/T | 8/T | 14/T | ∞ | ∞ | T | Sy L H |
| 3 | 0/- | 4/Sh | 14/T | 8/T | 14/T | 16/Sy | ∞ | Sy | L H A |
| 4 | 0/- | 4/Sh | 14/T | 8/T | 14/T | 16/Sy | ∞ | L | H A |
| 5 | 0/- | 4/Sh | 14/T | 8/T | 14/T | 16/Sy | ∞ | H | A |
| 6 | 0/- | 4/Sh | 14/T | 8/T | 14/T | 16/Sy | 20/A | A | R |
| 7 | 0/- | 4/Sh | 14/T | 8/T | 14/T | 16/Sy | 20/A | R |  |