CS201 Lab6 Report

SAIYAM BHARARA
2019EEB1189

DIFFERENT TYPES OF HEAPS

1) **Array based implementation**:- In this approach, we have used an array(vector) to maintain and find the minimum value node. The simple and naive approach of traversing the entire array to find the minimum value has been used. The time complexity to find the minimum element is O(n)(size of the vector or array). In this case, the cost of insertion is O(1). In the array based implementation of dijkstra, the overall complexity is O(V^2) where V is the total number of vertices in the graph. The complexity of bellman ford algorithm is O($V * E$). E is the number of edges in the given graph. As a result of this, the complexity of Johnson's algorithm becomes O($V^3 + V * E$).

2) **Binary heap based implementation**:- It is a faster and better approach in comparison to the array based. The time complexity of binary implemented dijkstra is O(($V + E$) * $log(V)$). The complexity of Bellman ford's algorithm remains unchanged. The overall complexity of Johnson's algorithm turns out to be O($V^2 + V * E * log(V) + V * E$). In Johnson's algo, dijkstra is run V times. The worst case complexity of insertion as well as deletion in a binary heap is O($log(n)$) where n is the total number of elements in the heap. Complexity for extract min and decrease key is also O($log(n)$).

3) **Binomial heap based implementation**:- This is a special kind of data structure which comprises trees having a total number of nodes as powers of 2. This heap has better performance in comparison to binary heap. Although it provides an efficient union operation of complexity O($log(n)$) as compared to O(n) in binary. The find min operation becomes a bit costly as compared to binary. Complexity for find min is O(1) in binary and O($log(n)$) in binomial. The complexity of Johnson's algorithm is($V^2 + V * E * log(V)$).

4) **Fibonacci heap based implementation**:- This data structure drastically reduces the complexity of functions, which are in common to other heaps. All the operations except extract min and delete have O(1) amortized time complexity. Extract min and delete operations have amortized complexity of O($log(n)$). The cost of find min is O(1) as we maintain a pointer to the minimum value of the heap. This data structure is implemented using circular doubly linked lists. The time complexity for fibonacci heap turns out to be O($V^2 * log(V) + V * E$).

Analysis of heaps on random large graphs

I have executed the code on random graphs in order to investigate the time complexities of various heap implementations.
V is the total number of nodes and E is the total number of edges. My code is showing segmentation fault for the fibonacci heap, therefore I am unable to do its analysis.

1) V=500 and E=1000, d=0
    a) Runtime for array implementation:- 0.137419 seconds
    b) Runtime for binary implementation:- 0.525863 seconds
    c) Runtime for binomial implementation:- 0.087316 seconds
2) V=1000 and E=1000, d=1
    a) Runtime for array implementation:- 0.208921 seconds
    b) Runtime for binary implementation:- 0.871062 seconds
    c) Runtime for binomial implementation:- 0.163516 seconds
3) V=600 and E=1000, d=1
    a) Runtime for array implementation:- 0.171687 seconds
    b) Runtime for binary implementation:- 0.662148 seconds
    c) Runtime for binomial implementation:- 0.110413 seconds
4) V=800 and E=1500, d=1
    a) Runtime for array implementation:- 0.576312 seconds
    b) Runtime for binary implementation:- 1.56948 seconds
    c) Runtime for binomial implementation:- 0.229613 seconds
5) V=1000 and E=1500, d=0
    a) Runtime for array implementation:- 0.797246 seconds
    b) Runtime for binary implementation:- 1.9071 seconds
    c) Runtime for binomial implementation:- 0.311607 seconds
6) V=500 and E=2000, d=0
    a) Runtime for array implementation:- 0.342086 seconds
    b) Runtime for binary implementation:- 0.97192 seconds

c) Runtime for binomial implementation:- 0.143574 seconds

7) V=600 and E=2000, d=1
   a) Runtime for array implementation:- 0.33308 seconds
   b) Runtime for binary implementation:- 1.21307 seconds
   c) Runtime for binomial implementation:- 0.181885 seconds
8) V=700 and E=2000, d=1
   a) Runtime for array implementation:- 0.435268 seconds
   b) Runtime for binary implementation:- 1.61068 seconds
   c) Runtime for binomial implementation:- 0.272647 seconds
9) V=1000 and E=1000, d=0
   a) Runtime for array implementation:- 0.377954 seconds
   b) Runtime for binary implementation:- 0.80326 seconds
   c) Runtime for binomial implementation:- 0.197553 seconds
10) V=600 and E=1500, d=0
   a) Runtime for array implementation:- 0.315259 seconds
   b) Runtime for binary implementation:- 0.83454 seconds
   c) Runtime for binomial implementation:- 0.151712 seconds

OBSERVATION

From the above results we can conclude that, binomial heap outperforms both binary and array based implementation of johnson's algorithm. Since my code for the fibonacci heap is not in working condition, I am unable to verify its complexity. Theoretically, a fibonacci heap has the best time complexity amongst all others. We know that the execution time of binary heap is better in comparison to the array based implementation. But in some cases as mentioned above, we can see that the binary heap is taking more time. This might be due to implementation differences and ambiguities.

CONCLUSION

From the above facts, we can deduce that for various different implementations of Johnson's algorithm, the time complexity is as follows

- **The Fibonacci heap** is the most efficient amongst all and has the least time complexity.
- Then comes the **binomial heap**, which is followed by the binary **heap.**
- The **array based implementation** is the least efficient.