Jake Goldstein

HW 11 Written

1. O(n + k)
2. Unsorted vector
   1. Insert: O(1) amortized
   2. Delete min: O(log(n))
   3. Find min: O(1)
3. [sentinel, 190, 380, 200, 388, 401, 277, 270, 1000, 399, 432]
   1. Representation

190

380 200

388 401 277 270

1000 399 432

* 1. Insert 197

190

197 200

388 380 277 270

1000 399 432 401

[sentinel, 190, 197, 200, 388, 380, 277, 270, 1000, 399, 432, 401]

1. Binary heap of sorts….
   1. [sentinel, 2, 3, 4, 6, 8, 70, 5, 9, 7, 19, 27]
   2. Remove root

3

6 4

7 8 70 5

9 27 19

1. Insert

template void BinaryHeap::insert( const Comparable & x ) {

if( theSize + 1 == array.size( ) )

array.resize( array.size( ) \* 2 + 1 );

// Percolate

++theSize;

int hole = theSize -1;

for( ; x < array[ hole / 2 ]; hole /= 2 )

array[ hole ] = std::move( array[ hole / 2 ] );

if(hole == 1 && x < array[0]){

array[hole] = std∷move(array[0]);

--hole;

}

array[ hole ] = x;

}

1. Input: 10, 12, 1, 14, 6, 5, 8, 15, 3, 7, 4, 11, 10, 0

[sentinel, 0, 3, 1, 6, 4, 10, 5, 15, 14, 12, 7, 11, 10, 8]

phase distance/predecessor visiting queue

G A B C D E F (front at queue on the left)

init 0/- G 1 0/- 1/G 1/G G F A

2 0/- 1/G 2/F 1/G F A E

Phase Distance / predecessor Visiting Queue

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | G | A | B | C | D | E | F |  |  |
| intial | 0/- |  |  |  |  |  |  |  | G |
| 1 | 0/- | 1/G |  |  |  |  | 1/G | G | F A |
| 2 | 0/- | 1/G |  |  |  | 2/F | 1/G | F | A E |
| 2 | 0/- | 1/G |  |  |  | 2/F | 1/G | A | E |
| 3 | 0/- | 1/G |  |  | 3/E | 2/F | 1/G | E | D |
| 4 | 0/- | 1/G | 4/D | 4/D | 3/E | 2/F | 1/G | D | B C |
| 5 | 0/- | 1/G | 4/D | 4/D | 3/E | 2/F | 1/G | B | C |
| 5 | 0/- | 1/G | 4/D | 4/D | 3/E | 2/F | 1/G | C |  |

Shortest path from G-C:

G F E D C

1. ­­

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G |
| A | - | 6 |  | 3 | 1 |  |  |
| B | 6 | - | 5 | 0 |  |  |  |
| C |  | 5 | - | 12 | 18 |  | 5 |
| D | 3 | 0 | 12 | - |  | 3 |  |
| E | 1 |  | 18 |  | - |  | 19 |
| F |  |  |  | 3 |  | - | 14 |
| G |  |  | 5 |  | 19 | 14 | - |

A –> D –> B –> E

B –> A –> D –> C

C –> B –> D –> G –>E

D –> A –> B –> C –>F

E –> A –> C –> G

F –> D –> G

G –> C –> F –> E

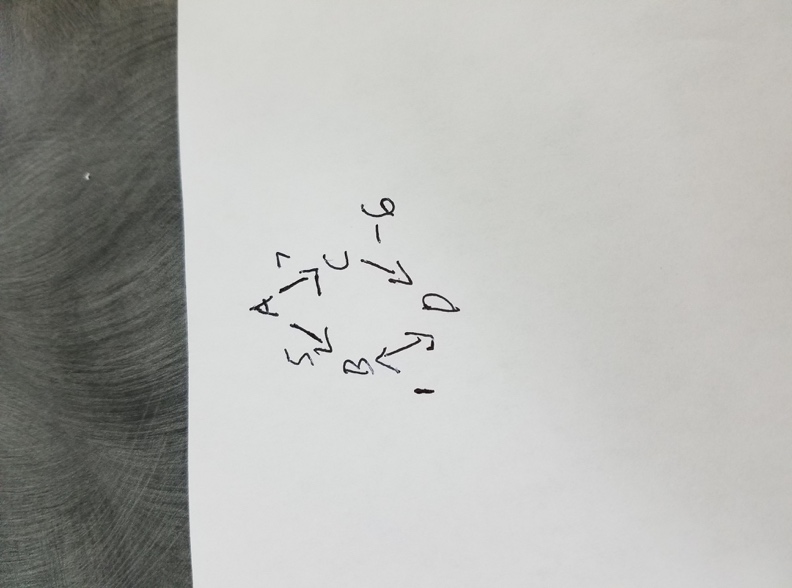
Phase Distance / predecessor

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | Visiting | discovered |
| initial | 0 | Inf | Inf | Inf | Inf | Inf | Inf |  | A |
| 1 | 0 | 6/A | Inf | 3/A | 1/A | Inf | Inf | A | B D E |
| 2 | 0 | 6/A | 19/E | 3/A | 1/A | Inf | 19/E | E | B C D G |
| 3 | 0 | 3/D | 15/D | 3/A | 1/A | 6/D | 19/E | D | B C F G |
| 4 | 0 | 3/D | 8/B | 3/A | 1/A | 6/D | 19/E | B | C F G |
| 5 | 0 | 3/D | 8/B | 3/A | 1/A | 6/D | 19/E | F | C G |
| 6 | 0 | 3/D | 8/B | 3/A | 1/A | 6/D | 13/C | C | G |
| 7 | 0 | 3/D | 8/B | 3/A | 1/A | 6/D | 13/C | G |  |

Shortest path from A to G:

A D B C G (13)

1. Wrong answer for Dijkstra’s Algorithm



1. In order to count how many of the same distance shortest paths there are I would create an unordered\_map where the key is the information about the vertex, and the value would be how many shortest paths there are to this vertex. Then every time that a new shortest path is found I would just find that element in my map, and add one to the value. Then if I wanted to know how many shortest paths there were I could just find that number. (This wouldn’t be able to tell me all of the shortest paths, but it would tell me that there are a certain number of them).