**Data Structure\_2071035 Lee Somin**

**Technical Report – prim\_algorithm.cpp**

*Theorical Explanation of Functions in ‘prim\_algorithm.cpp’*

**#define SWAP**

This definition executes swap operation between two data x and y, and use t for the temporary space.

**#define TRUE**

This definition represents true value, which is 1.

**#define FALSE**

This definition represents false value, which is 0.

**#define MAX\_VERTICES**

This definition represents the max number of vertices.

**#define INF**

This definition represents infinite, for meaning no connection between vetices and the initialization of dist value for each vertex.

**typedef struct vertex**

This structure contains the information about a vertex.

int selected – weather the vertex is selected

int dist – the distance value of the vertex

int parent – the index of the parent vertex

int child[] – the array of the index of the child vertices

int child\_num – the number of child that the vertex has

int heap\_id – the location index of the vertex in heap

**int weight[MAX\_VERTICES][ MAX\_VERTICES]**

This 2D array contains the edge information between vertices. INF means that there is no connection between the two vertices.

**vertex vertices[MAX\_VERTICES]**

This array contains each vertex of the graph.

**int minheap[MAX\_VERTICES+1]**

This array is for minheap based on the value of ‘dist’ for each vertex, and the array contains the id of each vertex.

**※ The heap starts from index 1, not 0.**

**int heap\_size**

This variable indicates the changed heap size due to the delete\_min\_heap() function.

**Build\_min\_heap**

Inputs: int\* A

Return: non

This function initializes the heap and executes the line 'inserts all vertices into the priority queue Q' in the pseudo code. Since all vertices except start vertex have INF for 'dist' at initial stage, this function just adds all vertices in to the heap in order of vertex id.

**Decrease\_key\_min\_heap**

Inputs: int\* A, int i, int key

Return: non

This function is for decreasing the value of the 'vertices[A[i]].dist' to given input 'int key'. If the key is smaller than the original key value, the function prints error message and returns. Since the heap is min heap, when key is decreased, the key value may be smaller than its ancestors. So, the function compares the value with its parent and if the key value of the parent is larger, the two nodes of the heap is swapped and i is updated to the index of swapped location. This iterates until the key changed is no more smaller than its parent.

**Increase\_key\_min\_heap**

Inputs: int\* A, int I, int key

Return: non

This function is for decreasing the value of the 'vertices[A[i]].dist' to given input 'int key'. If the key is bigger than the original key value, the function prints error message and returns. Since the heap is min heap, when key is increased, the key value may be bigger than its children. If the key value of the node is bigger than its child, the function swaps the two value's location. Unlike the decrease key, the swapped child can be bigger than its original sibling, which is child after swap. So, the function calls itself recursively for the swapped value. In this step, ‘A[i]--' is used since the function prints error is the value to change is equal to the original key. This sequence is executed iteratively until the heap qualifies the minheap property.

**delete\_min\_heap**

Inputs: int\* A

Return: int return\_id

This function is for deleting the top element in the heap. It sets return value(int return\_id) as the first node in heap. Then it swaps the first node of heap with the last node of the heap, then calls Increase\_key\_min\_heap to organize the nodes to qualify the heap property. ‘vertices[A[1]].dist--’ is for executing the ‘Increase\_key\_min\_heap()’ since it prints error message when the changed key is same as the original key. Lastly, it returns the return value, which is the vertex id of the original first node.

**prim**

Inputs: int s, int n

Return: non

This function executes the prim algorithm. First, it initializes distance and selected of all vertices to INF and FALSE. Then, the start vertex vertices[s]'s distance is set to 0 and the min heap is built. Inside the 'for' loop, the vertex with smallest dist value is selected by calling 'delete\_min\_heap()'. If the dist value of the selected vertex is INF, it means that it has no connection with other nodes, so the function returns. Otherwise, the inner for loop is started, and it updates the weight of the nonselected neighboring(the vertex that does not have INF for the weight) vertex's dist value to the smallest by calling 'Decrease\_key\_min\_heap()' and updates the parent of the vertex. When every vertex is selected, the function updates the child relationship of vertices by saving the child vertex id to the 'child[]' array.

**print\_prim**

Inputs: int s

Return: non

This function prints the prim algorithm tree starting from vertices[s] in preorder. For every child of the vertex, it calls 'print\_prim()' recursively. If the vertex is not starting vertex, the function prints out the information about the vertex.

**main**

In main function, prim algorithm is executed by calling 'prim()', then the result is printed by 'print\_prim()'.

**Result:**

텍스트이(가) 표시된 사진

자동 생성된 설명