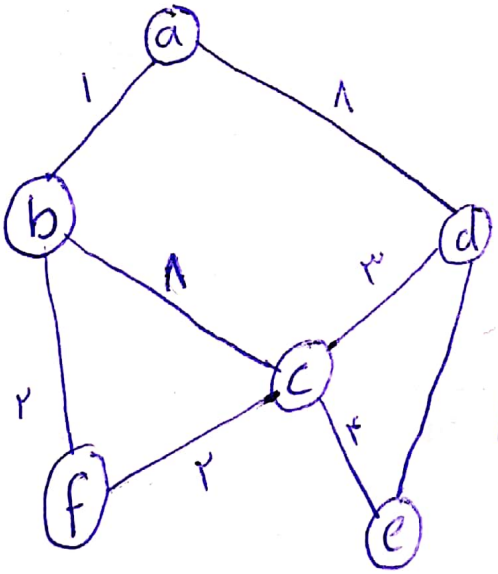
```
// this implementation is from https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5
// but comments are added by myself .
// this function is for finding the unvisited (it is not in mst set yet)
// vertex with minimum key (Key values used to pick minimum weight edge in cut )
int minKey(int key[], bool mstSet[]){
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (mstSet[v] == false && key[v] < min)// we could use min heap for better performance
            min = key[v], min_index = v;
    return min_index;
}
void primMST(int G[V][V]){
    int parent[V]; // hold the previous vertx of the vertex in mst set
    int key[V];
    bool mstSet[V];
    for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = false; // Initialize all keys as INFINITE
    key[0] = 0;
    parent[0] = -1; // First node is always root of MST
    for (int count = 0; count < V - 1; count++){
        int u = minKey(key, mstSet);
        mstSet[u] = true; // adding picked vertex to mst set
        // checking all neighbors of selected vertex and if the neighbor is not already in mst set
        // and the weight of edge is smaller than key of vertex then we update
        // its key and previous vertex .
        for (int v = 0; v < V; v++)
            if (G[u][v] && mstSet[v] == false && G[u][v] < key[v])
                parent[v] = u, key[v] = G[u][v];
    }
    // vertices in mstSet and their previous vertex
    // in parent set form all edges for mst
}
```

بخش الف

سوال ④ - بخش ب)

(این سوال کمی ابهام برانگیز بود ، امیدوارم درست متوجه شده باشم)

می دانیم که MST های مختلف بدست آمده ، همگی cost یکسانی خواهند داشت و همچنین تعداد این mst ها را انتخاب دو edge با وزن یکسان رخ می دهد برای مثال در گراف زیر ۴ mst متفاوت از انتخاب ۱ از ۲ بین یال های (bf , fc) و یال های (de , ce) داریم - راه حل ما می تواند این باشد که :



بعد از مرتب کردن یال ها به صورت صعودی ، هرکجا که دنباله ای از یال ها به صورت یکسان داریم ، (وزن یکسان)

یال هایی که در درخت T ما هستند را در index کمتر قرار می دهیم .

برای مثال اگر آرایه مرتب شده ما به صورت ab, bf, fc, cd, ce, de, ad, bc باشد ، اگر در T مورد نظر ما bf وجود داشت که اوکی است ، اما اگر به جای bf ، fc قرار داشت ما جای fc , bf را عوض می کنیم تا fc در index کوچک تر قرار بگیرد و زودتر در الگوریتم کروسکال بررسی شود .

چند حالت از این مثال :

اگر T ما شامل de و fc بود ← ab, fc, bf, cd, de, ce, ab, bc

جایشان عوض شد. جایشان عوض شد.

اگر T ما شامل de و bf بود ← ab, bf, fc, cd, de, ce, ab, bc