資料結構 第十一次作業

tags: 資料結構

資工112 張心瑜 40847044S

Shortest path problem

輸入資料

- 1. 第一行為三個正整數V、E、及C以空白區隔，分別代表城市的數目、公路的數目、以及州政府的數目。V個城市的編號對應正整數1到V，C個州政府的編號對應正整數1到C。(1 ≤ V ≤ 100、1 ≤ E ≤ 2000、1 ≤ C ≤ 20)
- 2. 第二行有V個介於1到C的正整數以空白區隔，表示每個城市所屬的州政府編號。
- 3. 接下來E行，每行有四個正整數以空白區隔，分別對應到公路的兩端城市編號(公路皆為雙向)、此段公路的通行費用P、以及公路所屬的州政府編號，1 ≤ P ≤ 50。
- 4. 下一行為由其他州政府所管轄公路所乘上的通行費倍率D，1 ≤ D ≤ 5。
- 5. 倒數第二行為一個正整數N，表示旅行的行程想停留的城市數目，1 ≤ N ≤ 10。
- 6. 最後一行是N個城市編號以空白區隔，表示行程依序想走訪的城市順序。

輸出資料

- 1.第一行書出最少總通行費用。

程式說明

用link list記兩城市之間出現超過兩條路

```
typedef struct road
{
    int32_t key;
    int32_t r_govenment;
    struct road *link;
}road;
```

選dist最小的值、回傳他的index

```
int choose (int32_t *distance, int32_t n, short int *found)
{   /* find the smallest distance not yet checked */
    int32_t min, minpos;
    min = 100000;
    minpos = -1;
    for (int32_t i = 0; i < n ; i++)
        if(distance[i] < min && !found[i])
        {
            min = distance[i];
            minpos = i;
        }
    return minpos;
}
```

尋找shortest path

```
for(int32_t i = 0 ; i < route_num-1 ; i++)
    {
        walk[cntwalk++] = route[i];
        short int found[v+1];
        int32_t dist[v+1];
        int32_t u;
        for(int32_t j = 0 ; j < v+1 ; j++)
        {
            found[j] = false;
            if(cost[route[i]][j].link == NULL)
            {
                if(city[route[i]] == cost[route[i]][j].r_govenment)
                    dist[j] = cost[route[i]][j].key;
                else if(cost[route[i]][j].r_govenment != -1 && city[route[i]] != cost[
                    dist[j] = cost[route[i]][j].key*money;
                else if(cost[route[i]][j].r_govenment == -1)
                    dist[j] = cost[route[i]][j].key;
            }
            else if(cost[route[i]][j].link != NULL)
            {
                int32_t tmp[2000];
                int32_t cnt = 0;
                struct road *w = &cost[route[i]][j];
                for( w ; w != NULL ; w = w->link)
                {
                    if(city[route[i]] == w->r_govenment)
                        tmp[cnt] = w->key;
                    else if(w->r_govenment != -1 && city[route[i]] != w->r_govenment)
                        tmp[cnt] = w->key*money;
                    else if(w->r_govenment == -1)
                        tmp[cnt] = w->key;
                    cnt ++;
                }
                int32_t min = tmp[0];
                for(int32_t k = 0 ; k < cnt ; k++)
                {
                    if(min > tmp[k])
                        min = tmp[k];
                }
                dist[j] = min;
            }
        }
        found[route[i]] = true;
        dist[route[i]] = 0;
        for(int32_t j = 0 ; j < v-1 ; j++)
        {
            u = choose(dist,v+1,found);
            found[u] = true;
            for(int32_t w = 1 ; w < v+1 ; w++)
            {
                if(!found[w])
                {
                    if(w == route[i+1])
                    {
```

```
            if(cost[u][w].link == NULL)
            {
                if(cost[u][w].r_govenment == city[route[i]])
                {
                    if(dist[u]+cost[u][w].key < dist[w])
                    {
                        dist[w] = dist[u] + cost[u][w].key;
                        walk[cntwalk++] =u;
                    }
                }
                else if(cost[u][w].r_govenment != city[route[i]])
                {
                    if(dist[u]+(cost[u][w].key*money) < dist[w])
                    {
                        dist[w] = dist[u] + (cost[u][w].key*money);
                        walk[cntwalk++] =u;
                    }
                }
            }
            else if(cost[u][w].link != NULL)
            {
                struct road *t = &cost[u][w];
                int32_t tmp[2000];
                int32_t cnt = 0;
                for( t ; t != NULL ; t = t->link)
                {
                    if(t->r_govenment == city[route[i]])
                    {
                        if(dist[u] + t->key < dist[w])
                        {
                            tmp[cnt++] = dist[u] + t->key;
                            walk[cntwalk++] = u;
                        }
                    }
                    else if(t->r_govenment != city[route[i]])
                    {
                        if(dist[u]+(t->key*money) < dist[w])
                        {
                            tmp[cnt++] = dist[u] + (t->key*money);
                            walk[cntwalk++] = u;
                        }
                    }
                }
                int32_t min = tmp[0];
                for(int32_t k = 0 ; k < cnt ; k++)
                {
                    if(min > tmp[k])
                        min = tmp[k];
                }
                if(cnt > 0)
                    dist[w] = min;
            }
        }
        else
        {
```

```
                    if(cost[u][w].link == NULL)
                    {
                        if(cost[u][w].r_govenment == city[route[i]])
                        {
                            if(dist[u]+cost[u][w].key < dist[w])
                                dist[w] = dist[u] + cost[u][w].key;
                        }
                        else if(cost[u][w].r_govenment != city[route[i]])
                        {
                            if(dist[u]+(cost[u][w].key*money) < dist[w])
                                dist[w] = dist[u] + (cost[u][w].key*money);
                        }
                    }
                    else if(cost[u][w].link != NULL)
                    {
                        struct road *t = &cost[u][w];
                        int32_t tmp[2000];
                        int32_t cnt = 0;
                        for( t ; t != NULL ; t = t->link)
                        {
                            if(t->r_govenment == city[route[i]])
                            {
                                if(dist[u] + t->key < dist[w])
                                    tmp[cnt++] = dist[u] + t->key;
                            }
                            else if(t->r_govenment != city[route[i]])
                            {
                                if(dist[u]+(t->key*money) < dist[w])
                                    tmp[cnt++] = dist[u] + (t->key*money);
                            }
                        }
                        int32_t min = tmp[0];
                        for(int32_t k = 0 ; k < cnt ; k++)
                        {
                            if(min > tmp[k])
                                min = tmp[k];
                        }
                        if(cnt > 0)
                            dist[w] = min;
                    }
                }
            }
        }
    }
```