最短路径之Dijkstra算法

姓名: 王恩策
学号: 12081228

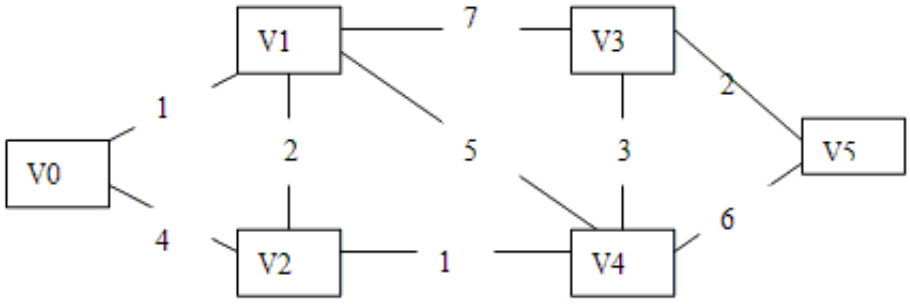
算法简介

Dijkstra算法是由荷兰计算机科学家艾兹赫尔·戴克斯特拉提出。该算法常用于路由算法或者作为其他图算法的一个子模块。举例来说，如果图中的顶点表示城市，而边上的权重表示著城市间开车行经的距离，该算法可以用来找到两个城市之间的最短路径。

开放最短路径优先算法是该算法在网络路由中的一个具体实现。

算法原理

实例分析：求图中V0到V5的最短路径



a.根据图来建立权值矩阵：

```
//代表无限大
#define inf     100000
//点的个数
#define V_SIZE    6
int W[V_SIZE][V_SIZE] = {
            {  0,   1,   4,  inf,  inf,  inf },
            {  1,   0,   2,   7,    5,  inf },
            {  4,   2,   0,  inf,    1,  inf },
            { inf,  7,  inf,   0,    3,    2 },
            { inf,  5,    1,   3,   0,    6 },
            { inf, inf,  inf,   2,   6,    0 }
           };
```

例如：

- `W[0][2]=4` 表示点V0到点V2的权值为4

- `W[0][3]=inf` 表示点V0与V3不相邻，所以权值无限大

b.算法流程

1. 对V0标号(即表示到V0的最短路径已找到)，取出V0到其它点的路径得到 `distance: {0,1,4,inf,inf,inf}` 找到V0到各点中权值最小的那个点（标号的点除外,inf代表无限大），故得到1即对应的下标1，得到V1；对V1标号，然后更改V0通过V1到其它点的路径(即V0->V1->Vn如果小于V0->Vn则采用新值)，得到 `distance: { 0, 1, 3, 8, 6, -1};`
2. 找到distance中权值最小的那个点，（标号的点除外）得到V2,对V2标号，然后更改V0通过V0->V2到其它点的路径得到 `distance: { 0, 1, 3, 8, 4, inf};`
3. 找到distance中权值最小的那个点，（标号的点除外）得到V4,对V4标号，然后更改V0通过V0->V4到其它点的路径得到 `distance: { 0, 1, 3, 7, 4, 10};`
4. 找到distance中权值最小的那个点，（标号的点除外）得到V3,对V3标号，然后更改V0通过V0->V3到其它点的路径得到 `distance: { 0, 1, 3, 7, 4, 9};`
5. 最后只剩下V5没有被标号，就找到V5了。

算法实现

C语言源代码：

```
/*
 * dijkstra_test.c
 *
 *  Created on: 2015年5月12日
 *      Author: wangence
 */
#include <stdio.h>
#include <stdlib.h>

//代表无限大
#define inf     100000
//点的个数
#define V_SIZE    6

//数组填充
void array_fill(int * array, int len, int val) {
    int i;
    for (i = 0; i < len; i++) {
        array[i] = val;
    }
}
/*函数功能:
 *     dijkstra算法求无向图最短距离，并输出路径
 *输入参数:
 *    graph     权值矩阵
 *    n         矩阵大小,即点的个数
 *    start     起点
 *    dist     接收到所有点的最短距离
 * */
```

```
void dijkstra(int graph[][V_SIZE],int n,int start,int dist[]) {
    int* path=(int*)malloc(sizeof(int)*n);
    int* shortest=(int*)malloc(sizeof(int)*n);
    int* mark=(int*)malloc(sizeof(int)*n);
    int min,v,i,j;
    array_fill(mark,n, 0);
    array_fill(dist,n, inf);

    for(i=0;i<n;i++) {
        dist[i]=graph[start][i];
        if(i!=start&&dist[i]<inf)path[i]=start;
        else path[i]=-1;
    }
    mark[start]=1;
    while(1) {
        min=inf;v=-1;
        //找到最小的距离
        for(i=0;i<n;i++) {
            if(!mark[i]) {
                if(dist[i]<min) {min=dist[i];v=i;}
            }
        }
        if(v==-1)break; //找不到更短的路径了
        //更新最短路径
        mark[v]=1;
        for(i=0;i<n;i++) {
            if(!mark[i]&&
                    graph[v][i]!=inf&&
                    dist[v]+graph[v][i]<dist[i]) {
                dist[i]=dist[v]+graph[v][i];
                path[i]=v;
            }
        }
    }
    //输出路径
    printf("起点\t终点\t最短距离\t对应路径 \n");
    for(i=0;i<n;i++) {
        if(i==start) continue;
        array_fill(shortest,n, 0);
        printf("%d\t",start);
        printf("%d\t",i);
        printf("%d\t",dist[i]);
        int k=0;
        shortest[k]=i;
        while(path[shortest[k]]!=start) {
            k++;shortest[k]=path[shortest[k-1]];
        }
        k++;shortest[k]=start;
        for(j=k;j>0;j--) {
            printf("%d->",shortest[j]);
        }
        printf("%d\n",shortest[0]);
    }
    free(path);
    free(shortest);
    free(mark);
    return ;
```

```
}

int main()
{
    int dist[V_SIZE];
    int W[V_SIZE][V_SIZE] = {
                { 0,   1,   4,  inf,  inf,  inf },
                { 1,   0,   2,   7,    5,  inf },
                { 4,   2,   0,  inf,   1,  inf },
                { inf, 7,  inf,  0,    3,    2 },
                { inf, 5,    1,  3,    0,    6 },
                { inf, inf,  inf,  2,    6,    0 }
               };

     dijkstra(W,V_SIZE,2,dist);
     return 0;
}
```

运行结果：

```
起点    终点    最短距离        对应路径
2       0       3               2->1->0
2       1       2               2->1
2       3       4               2->4->3
2       4       1               2->4
2       5       6               2->4->3->5
```

May 12 2015 9:19 AM