中山大学数据科学与计算机学院
程序设计与数据结构综合实践 II
TSP 问题实验报告

（2018-2019 学年秋季学期）

学　　号：16337113
姓　　名：劳马东
专　　业：超算

一、实验题目

完成下列三项任务：

- 实现一个高效的无向简单图的 TSP 算法，在小规模内有精确最优解，对较大的规模能有近似解；
- 生成求解输入，不仅可以用于测试自己的程序，也能给其他组的同学出题，比较求解速度和结果精确度；
- 检验输出是否合法，是否能在图中完成 TSP 任务，并且路程计算无误。

输入：第一行为 n，接下来是 n 乘 n 的矩阵，表示两点之间的距离，示例如下：

$$\begin{array}{cccc} 4 & & & \\ 0 & 1 & 2 & 3 \\ 1 & 0 & 4 & 5 \\ 2 & 4 & 0 & 6 \\ 3 & 5 & 6 & 0 \end{array}$$

输出：第一行为解的总路程，第二行为路径过程。如上图可给出的解为：

$$\begin{array}{ccccc} 14 & & & & \\ 0 & 1 & 2 & 3 & 0 \end{array}$$

表示总路程为 14(1+4+6+3)，路径为 0->1->2->3->0。

二、实验内容

（一）分支定界算法求解 TSP 精确解

分支定界算法（branch and bound）是一种有信息搜索算法。每个节点的估计值由两部分组成：节点的实际代价和启发值，即：

$$f(x) = g(x) + h(x) \tag{1}$$

其中 $g(x)$ 是节点的当前实际路径消耗，$h(x)$ 是节点 x 到终点的启发值。

假设邻接矩阵的权值如下：

$$C(i,j) = \begin{cases} W(i,j) & \text{i != j} \\ INFINITY & \text{i == j} \end{cases} \tag{2}$$

例如图1的邻接矩阵 M。

	C0	C1	C2	C3	C4
C0	INF	20	30	10	11
C1	15	INF	16	4	2
C2	3	5	INF	2	4
C3	19	6	18	INF	3
C4	16	4	7	16	INF

图 1. CSP 完全图示例

有信息搜索算法（如 A* 和 IDA*）的细节不再细写，在《人工智能》课程上已经有了很好的理解。下面主要讲算法的启发函数，如何计算一个节点的估计代价呢？使用下面的方法：对于邻接矩阵的每一行、每一列，找到对应行、列的最小值，节点的启发函数值就是这些最小值的和，即：

$$g(x) = \sum_{i=1}^{N}(min(row_i) + min(column_i)) \tag{3}$$

。

从父节点到子节点的行动如何表示呢？例如，要从城市 i 移动到城市 j，邻接矩阵如何改变？首先，我们把城市 i 的所有出边置为 INFINITY，把城市 j 的所有入边置为 INFINITY，把城市 j 到城市 0 的边置为 INFINITY。然后，对矩阵的每一行、每一列，执行一个松弛操作：减去对应的最小值，最终使每一行、每一列有至少一个值为 0。例如，从城市 0 移动到城市 1，图2(b)中红色标注的元素被置为 INFINITY，变成图2(b)。除此之外，行动会导致邻接矩阵被松弛：每一行、每一列减去对应的最小值，最终导致每一行、列至少有一个权值为 0。

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

(a) 父节点 M

INF	INF	INF	INF	INF
INF	INF	11	2	0
0	INF	INF	0	2
15	INF	12	INF	0
11	INF	0	12	INF

(b) 子节点 M

图 2. 置 INFINITY 示例

算法 1 节点间的行动

输入： $from, to$：城市编号，M：城市 $from$ 的 $n \times n$ 邻接矩阵

```
function MOVE(from, to, M)
    n ← M.size()
    for i ∈ [0, n) do
        M[from][i] ← M[i][to] ← INFINITY
    end for
    M[to][0] ← INFINITY
    reduceRow(M, n)
    reduceCol(M, n)
end function
```

（二）生成输入

城市与城市之间的权值随机确定，由于城市到自己没有路径，而且问题是针对无向图的，因此需要使用一个二维矩阵来存储随机结果，以保证 $M[i][j] == M[j][i]$。

算法 2 A* 算法

```
function MOVE(M)
    heap：以节点估计值存储的最小堆
    root ← Node(0)
    root.cost ← heuristic(M)
    move(−1, 0, M)
    root.matrix ← M
    heap.push(root)
    while heap is not empty do
        best ← heap.pop()
        for each neighbor v of best do
            new_node ← Node(v)
            M_tmp ← copy(best.matrix)
            new_node.cost ← best.cost + M_tmp[best.num][v]+ heuristic(M_tmp)
            move(best.num, v, M_tmp)
            new_node.matrix ← M_tmp
            heap.push(new_node)
        end for
    end while
end function
```

（三）输出检验

检验包括四个方面：输出结果是否涵盖所有城市？是不是每个城市只经过 1 次？顺着输出结果能不能走到起点？路径代价的精确度如何？

对于第一个问题，使用一个全集，每经过一个城市就从集合中删除一个城市，最终集合为空时假设成立。

对于第二个问题，使用一个统计数组，如果发现某个元素不等于 1，假设不成立。

对于第三个问题，顺着输出结果在图上走，如果发现边不合法，输出错误；结果的最后一个值必须与第一个值相同。

上一步可以统计路径的权值，将其和暴力搜索的结果相比，判断是否相等（需要精确解时）或计算精度（需要近似解时）。

三、关键代码

1、 分支定界算法 A* 部分

```
pair<int, vector<int>> solve(vector<vector<int>>& cost_matrix) {
    ...
    Node *root = new Node(cost_matrix, path, 0, -1, 0);
    pq.push(root);    // pq是以节点的lower_bound值排序的最小堆
    while (!pq.empty()) {
        Node *best = pq.top();    // 取当前f(x)最小的一个
        pq.pop();
        if (best->level == n - 1) // 找到解
        else {
            for (int j = 0; j < n; ++j) {
                int w = best->reduced_matrix[best->vertex][j];
                if (w != INF) {
                    // 扩展节点，由于做了松弛，总的估计值还需加上父节点启发值
                    Node *child = new Node(best->reduced_matrix, best->path,
                            best->vertex, j, best->lower_bound + w);
                    pq.push(child);
                }
            }
        }
    }
    ...
}
```

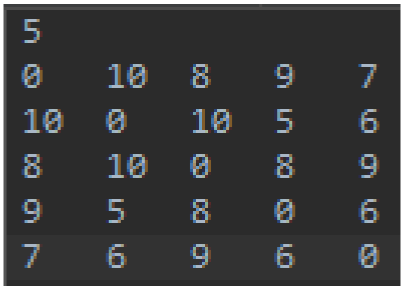
代码清单 1. A*

2、 分支定界算法计算启发值与松弛

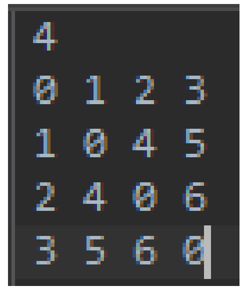
```
Node(vector<vector<int>> m, vector<pair<int, int>> p,
     int from, int to, int g=0): path(p), reduced_matrix(m), vertex(j) {
    ...
    if (from > -1) {      // 第from行、to列置为INF
        for (int k = 0; k < n; ++k) {
            reduced_matrix[from][k] = reduced_matrix[k][to] = INF;
        }
    }
    reduced_matrix[to][0] = INF;    // 节点到起点的边置为INF
    // heuristic计算启发值，然后对行和列执行reduce操作
    lower_bound = g + heuristic(reduced_matrix);
}
```

代码清单 2. 创建节点

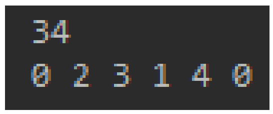
四、实验结果



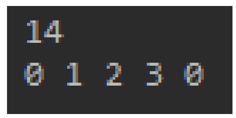
(a) 输入 1



(b) 输入 2



(c) 输出 1



(d) 输出 2

n	时间/s
5	0
10	0.013
15	0.736
20	21.836

表 1. 运行时间