5-5：

> *设 G 是有 n 个结点的有向图，从顶点 i 发出的边的最大费用记为 $max(i)$。*
>
> *(1) 证明旅行售货员回路的费用不超过 $\sum_{i=1}^{n} max(i)+1$。*
>
> *(2) 在旅行售货员问题的回溯法中，用上面的界作为 $bestc$ 的初始值，重写该算法，并尽可能地简化代码*

题目分析

本题共有两小题，对于(1)，易证如下：

证：假设结论不成立，即 $cost \geq \sum_{i=1}^{n} max(i)+1$。现考虑回路的最大费用：如果旅行售货员在走回路的过程中，每次都走最大费用的回路，即每次从顶点 i 出发到下一个顶点，均走花费为 $max(i)$ 的边，则此时回路的总花费为：$\sum_{i=1}^{n} max(i)$，与假设矛盾，因此结论成立，即：

旅行售货员回路的费用不超过 $\sum_{i=1}^{n} max(i)+1$，

或可简化表示为：$cost < \sum_{i=1}^{n} max(i)+1$。

Q.E.D.

(2)：

对于本题，只需要修改原回溯法算法即可。

代码实现

注：本算法使用c++语言进行重构，需要用到的变量将在算法顶部列出并给予注释，下面的代码仅包含traceback部分，对于整个程序，初始化图部分与 5.6 题使用相同的init()函数，同时使用与5.6中相同的头文件与main函数以方便对5.5，5.6以及原算法进行比较。

```
int n;  // 图G的结点数
int x[100];  // 当前的解空间
int bestx[100];  // 当前的最优解
float bestcost;  // 当前最优值
float ccost;  // 当前费用
float a[100][100];  // 图G的邻接矩阵
float maxcost[100];  // 对于每一个结点i，其最大出边的cost为maxcost[i]
float MAXCOST;  // 用于记录剩余结点的 maxcost之和，一开始初始化为maxcost[i]之和
const float MAX = 2000000.0;  // 无穷大，表示没有路径相连
void backtrace(int i) {
    if (i == n) {
        if (a[x[n-1]][x[n]] < MAX && a[x[n]][1] < MAX &&
            ccost + a[x[n-1]][x[n]] + a[x[n]][1] < bestcost) {
```

```
            // 这里省略了 bestcost == MAX，因为bestcost初始化已修改
            for (int j = 1; j <= n; j++) bestx[j] = x[j];
            bestcost = ccost + a[x[n - 1]][x[n]] + a[x[n]][1];
        }
    } // 对于i==n的部分可以进行优化
    else {
        for(int j = i; j <= n; j++) {
            if (a[x[i - 1]][x[j]] < MAX && ccost + a[x[i - 1]]
[x[j]] < bestcost) {
                // 此处的条件判断也可以进行优化
                swap(x[i], x[j]);
                ccost += a[x[i - 1]][x[i]];
                backtrace(i + 1);
                Sleep(2);  // 每次回溯一次就要休眠2ms，为了对比不同算
法之间的效率
                ccost -= a[x[i - 1]][x[i]];
                swap(x[i], x[j]);
            }
        }
    }
}
```

对于回溯算法 $backtrace$，主要修改的地方就是去掉了条件判断中的 $bestcost == MAX$，因为 $bestcost$ 已经初始化了。

样例输出

样例于init()函数内初始化，本样例的输出为：

```
最短路径长为：39
最短路径为：1 7 6 3 10 2 4 5 8 9
回溯耗时为：4349ms
```

而本样例也有其他合理的路径，比如1->3->8->7->6->5->4->2->10->9->1，其花费为：6+9+5+1+9+2+5+8+6+5=56；也有1->3->6->7->2->10->4->5->8->9，总花费为44，花费显然比最优解大。回溯耗时为4349ms，在5-6中将于原版与新的上界函数进行对比。

算法分析

与 $backtrace$ 算法原本的 $O(n!)$ 相比，修改后的回溯算法增加的时间复杂度可以认为是 $O(1)$ 的，因此总复杂度仍然为 $O(n!)$。