EE3980 Algorithms

HW5 Ranking Martial Artists

104061212 馮立俞

2018/4/8

Introduction

In this assignment, we are given 108 martial artist names and 63*5 = 315 1 vs. 1 match results. Then, we're required to rank the martial artists according to the matches.

Approach

The match result can be represented by a Directed Acyclic Graph(DAG) in which the edges point from match winner to match losers. Therefore, the graph consists of 108 vertices and 315 edges, which is too wasteful to construct the graph using adjacency matrix. As a result, I chose to build the graph using linked lists.

After the graph is constructed, we can sort the vertices using topology sort. It's noteworthy that since the graph is sparse, more than one valid sorting results are valid.

Topology Sort

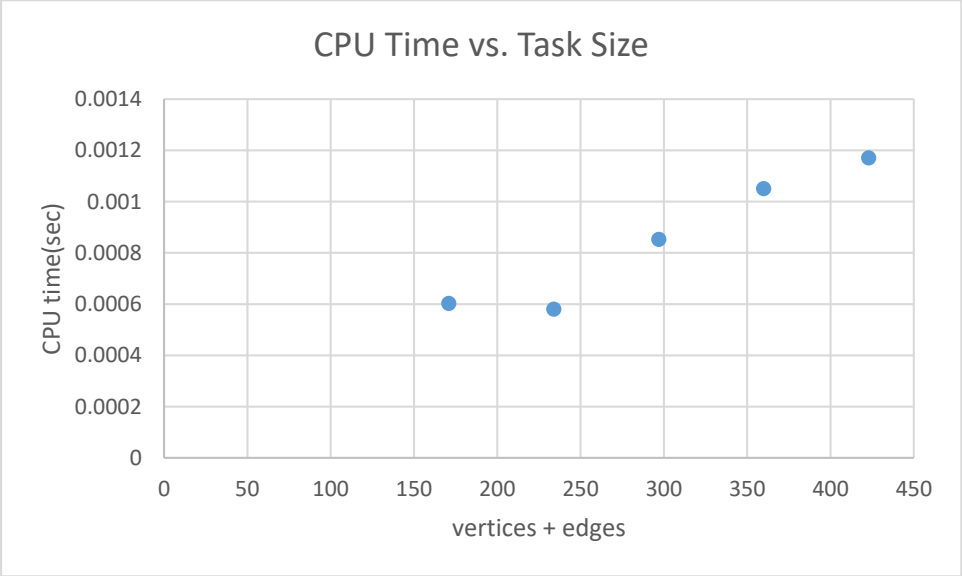
```
1.  Algorithm top_sort(v, slist)
                        // Topological sort using depth first search algorithm.
2.
    // v is the vertex being visited; and slist is the ordered linked list.
3.  {
4.          visited[v]: = 1;
5.          for each vertex w adjacent to v do {
6.              if (visited[w] = 0) then top_sort(w);
7.          }
8.          add v to the head of slist;
9.
10. }
11. Algorithm topsort_Call(v) // Initialization and recursive top_sort function call
    .
12.  {
13.
14.         for v: = 1 to n do visited[v]: = 0;
15.         slist: = NULL;
16.
17.         for v: = 1 to n do
18.             if (visited[v] = 0) then top_sort(v, slist);
19.
20.  }
```

In this algorithm, we use Depth First Search(DFS) to traverse the vertices. Since linked list is adopted, the complexity of traversal is $O(n + e)$, where n and e are the number of vertices and edges in the graph. If we use adjacency matrix, the complexity could grow to $O(n^2)$. Also, the space complexity of two data structure are $O(n + e)$ and $O(n^2)$, respectively.

Results and analysis

Efficiency

We can plot the overall execution time w.r.t. $n + e$ as follows



The curve is quite linear as e grows linearly. The abnormal behavior of the second point might be caused by initializing overhead.

Correctness

Since there's more than one valid output, I didn't check the output deterministically. Rather, I picked out some edges, then observed if the output obeys them. So far they're valid.

Observations and Conclusion

Though being a linear complexity algorithm, the execution time of linked list approach seem to be much slower when it's compared to other sorting algorithms in the previous assignments. This could result from the implemented data structure.

```c
/*********************************************
EE3980 HW05 Ranking Martial Artists
104061212  Li-Yu Feng
Date:2018/4/8
*********************************************/
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>


typedef struct node{            //graph node
    char *name;
    int index;
    bool visited;
    struct node *next, *end;
}Node;

int insertNode(Node **list, int N, char *winner, char *loser);  //record match

void top_sort(int index, Node **list, char **rank);             //topology sort

void rankit(char *name, char **rank);                           //store result
double GetTime(void);

int insertNode(Node **list, int N, char *winner, char *loser){
    Node *temp, *temp2;
    int i;      //looping index
    int j, k;   //to find winner,loser
    int m, n;   //record winnner/loser location

    i = 0;
    j = 1;
    k = 1;
    m = -1;
    n = -1;
    for(i = 0, j = 1; i < N && j != 0; i++){    //find winner location
        j = strcmp(list[i]->name, winner);
    }
    m = i-1;
    for(i = 0, k = 1; i < N && k!=0; i++){      //find loser
        k = strcmp(list[i]->name, loser);
    }
    n = i-1;

    temp = list[m]->end;

    temp2 = malloc(sizeof(Node));
    temp2->name = malloc(  strlen(loser) + 1 );
    temp2->name = loser;
```

```
49      temp2->visited = false;             //
50      temp2->index = n;                   //
51      temp2->next = NULL;                 //add node to linked list
52      temp->next = temp2;
53      list[m]->end = temp2;
54      return 0;
55  }
56
57  void top_sort(int index, Node **list, char **rank){
58      Node *temp = list[index];
59
60      temp->visited = true;
61      for(;temp != NULL; temp = temp->next)
62          if(list[temp->index]->visited == false)
63              top_sort(temp->index,list,rank);
64      rankit(list[index]->name,rank);
65  }
66
67  void rankit(char *name, char ** rank){
68      static int i= 107;      //Nplayers = 108
69
70      rank[i--] = name;
71  }
72
73  double GetTime(void)
74  {
75      struct timeval tv;
76      gettimeofday(&tv,NULL);
77      return tv.tv_sec+1e-6*tv.tv_usec;
78  }
79
80  int main(){
81
82      int Nplayers,Ntour;     //108,63
83      Node **NameList, *iter;
84      int i,j,k;
85      char *temp1, *temp2;    //input buffer
86      char **rank;            //final ranking
87      double time;
88
89      temp1 = malloc(sizeof(char *));
90      temp2 = malloc(sizeof(char *));
91
92      scanf("%d", &Nplayers);
93      printf("%d\n",Nplayers );
94
95
96      NameList = (Node **)malloc(Nplayers * sizeof(Node *));
97      for (i = 0; i < Nplayers; ++i){
98          NameList[i] = (Node *)malloc(sizeof(Node *));
```

```
 99          NameList[i]->name = malloc(sizeof(char *));
100      }
101
102      time  = GetTime();
103      for (i = 0; i < Nplayers; ++i)
104      {
105          scanf("%s", NameList[i]->name);         //read martial artists' name
106          NameList[i]->end = NameList[i];
107          NameList[i]->next = NULL;
108          NameList[i]->visited = false;                       //input players &
109          NameList[i]->index = i;                             //init NameList
110      }
111
112      rank = (char **)malloc(Nplayers * sizeof(char *));      //
113      for (i = 0; i < Nplayers; ++i){                         //
114          rank[i] = (char *)malloc(sizeof(char *));           //initialize rank
115      }
116
117      for(j = 0; j < 5; j++){             //tour1~tour5
118          scanf("%d", &Ntour);
119          for (i = 0; i < Ntour; ++i){
120              scanf("%s %s %s",temp1, temp2, temp2 );
121              insertNode(NameList, Nplayers, temp1, temp2);
122          }
123      }
124
125      for(i = 0; i < Nplayers; i++){          //call topology sort
126          if( NameList[i]->visited ==false )
127              top_sort(i, NameList, rank);
128      }
129
130      for(i = 0; i < Nplayers; i++){          //print ranking
131          printf("%d:%s\n",i+1, rank[i]);
132      }
133      printf("CPU Time = %.3g sec\n",GetTime()-time );
134
135      return 0;
136  }
```

Score: 40

———-

[Late turn in] on 4/9 00:46

[Compiler warnings] line 76.
- All compiler warnings should be resolved before turning in.

[Program output] is incorrect.