TIMUS 1806 Mobile Telegraphs Solution

write by Gestalti Lur
2012-07-15

题目大意

有 N(2<=N<=50000)个长度为 10 的数字串{s}。数字串 s_i,s_j(i != j)是相互可达的当且仅当其 s_i 改变一个数字后和 s_j 相同或者其交换某两个位置上的数字之后和 s_j 相同，保证所有的 s 都是不同的。s_i 到 s_j 转换的权值为其最长公共前缀的长度 k 所对应的权值 v_k。{v}会在输入数据中给出，为一个长度为 10 的不增序列。问数字串 1 到 N 是否可以通过这样的转换达到，若能则输出其最小权值和其通过的路径，否则输出“-1”。

算法分析

首先可以想到因为 N 有 10^4 而不能直接建图。其次 s_i 到 s_j 可以转换意味着 s_i 自身改变一个数字或者交换两个数字之后和 s_j 相同。设 s_i 改变或者交换操作之后生成的是 t,那么如何知道 t 是否在{s}中。显然可以用 hash 来解决。另外还需要考虑到 s_i 如果改变位置 k 的数字后和 s_j 相同，那么它们的最长公共前缀显然是 k-1,同理如果交换 k1,k2 位置上的数字后相同，那么则为 min(k1,k2)-1。所以其最长公共前缀也可以用 O(1)的时间复杂度算出来。这样如果使用 heap+dijkstra 算法（因为 dijkstra 算法每个串只会入堆一次），所需要的时间复杂度大致为 O((10*10+10)*N*log(N)*p),log(N)为 heap 操作的时间复杂度，p 为 hash 的期望值。如果使用迭代写法的 heap 则要更快一些。

参考代码

```
/*
  TIMUS 1806
  2012-07-11
  ACCEPTED
  write by gestapolur
*/
#include<iostream>
#include<cstring>
#include<vector>
#include<queue>
#define MAXN 100005
#define MAXH 6000005
#define INF 1 << 30
using namespace std;

class node
{
public:
  int dist , idx;
  node ( int dist , int idx ) : dist( dist ) , idx( idx ) {}
};

class cmp
{
```

```
public:
  bool operator () ( const node a , const node b ) const
  { return ( a.dist > b.dist ); }
};

int n , cnt;
int val[ 11 ] , mark[ MAXN ] , pre[ MAXN ];
priority_queue< node , vector<node> , cmp > min_heap;
char s[ MAXN ][ 11 ];
bool ins[ MAXN ];

int head[ MAXH ] , Next[ MAXH ];//hash

int sv[ MAXN ];//saved path

inline int Hash( char *str )
{
  int v = 0 , seed = 131;
  while( *str )
    v = v * seed + *(str ++ );
  return ( v & 0x7fffffff ) & MAXH;
}

inline void insert_Hash( int idx )
{
  int i , h;
  h = Hash( s[ idx ] );
  Next[ idx ] = head[ h ];
  head[ h ] = idx;
}

void init()
{
  int i , j;
  char tmp[ MAXN ];
  cin>>n;
  for( i = 1 ; i <= 10 ; ++ i )
    cin>>val[ i ];
  for( i = 1 ; i <= n ; ++ i )
    cin>>s[ i ];

  memset( head , -1 , sizeof( head ) );
  memset( Next , -1 , sizeof( Next ) );

  for( i = 1 ; i <= n ; ++ i )
    insert_Hash( i );

  return ;
```

```
}

void out()
{
 if( mark[ n ] == INF )
   {
    cout<<"-1\n";
    return ;
   }
 cout<<mark[ n ]<<"\n";
 int cur = n;
 do{
  sv[ ++ cnt ] = cur;
  cur = pre[ cur ];
 }while( cur );
 cout<<cnt<<"\n";
 do{
  cout<<sv[ cnt -- ];
  if( cnt ) cout<<" ";
  else cout<<"\n";
 }while( cnt );

 return ;
}

inline void relax( char *str , int dist , int idx , int lcp )
{
 int tcode = Hash( str );
 int x = head[ tcode ];
 while( x not_eq -1 )
   {
    if( not strcmp( s[ x ] , str ) )
    {
      int dval = val[ lcp ];
      //cout<<idx<<"->"<<x<<" "<<str<<" "<<dval<<" "<<mark[ x ]<<"\n";
      if( mark[ x ] > dist + dval )
        {
          mark[ x ] = dist + dval;
          pre[ x ] = idx;
          min_heap.push( node( mark[ x ] , x ) );
        }
      return ;
    }
    x = Next[ x ];
   }
 return ;
}
```

```
void dijkstra()
{
 int i , j , mdist , midx , vist;
 char str[ 11 ], tmp;

 for( i = 1 ; i <= n ; ++ i )
  mark[ i ] = INF;
 vist = 0;
 mark[ 1 ] = 0;
 min_heap.push( node( 0 , 1 ) );

 do{
  do{
   mdist = min_heap.top().dist;
   midx = min_heap.top().idx;
   min_heap.pop();
  }while( not min_heap.empty() and ins[ midx ] );

  ins[ midx ] = true;
  strcpy( str , s[ midx ] );

  //relax
  for( i = 0 ; i < 10 ; ++ i )
   {
   for( j = i + 1 ; j < 10 ; ++ j )
    if( str[ i ] not_eq str[ j ] )
     {
      //swap
      tmp = str[ i ]; str[ i ] = str[ j ]; str[ j ] = tmp;
      relax( str , mark[ midx ] , midx , i + 1 );
      tmp = str[ i ]; str[ i ] = str[ j ]; str[ j ] = tmp;
     }
   for( j = 0 ; j < 10 ; ++ j )
    if( char( j + 48 ) not_eq str[ i ] )
     {
      //change
      tmp = str[ i ]; str[ i ] = char( j + 48 );
      relax( str , mark[ midx ] , midx , i + 1 );
      str[ i ] = tmp;
     }
   }
  ++ vist;
 }while( vist < n and not min_heap.empty() );
 return ;
}

int main()
{
```

```
  init();

  dijkstra();

  out();

  return 0;
}
```