[CSE3081(2 반)] 알고리즘 설계와 분석 과제 3 보고서

20151623 한상구

1. Minimal Triangulation, Subset sum 각각에 대한 설명
   a. Minimal Triangulation

      점화식은 밑과 같이 구성했습니다.

$$D_{is} = \begin{cases} 0, & if\ 0 \le s < 4 \\ min_{1 \le k \le s-2}\left[D_{i,k+1} + D_{i+k,s-k} + W_{i,i+k,i+s-1}\right], & if\ s \ge 4 \\ where\ W_{i,i+k,i+s-1}\ is\ length\ of\ chords & \end{cases}$$

      인접한 Vertex 간의 length 는 0 으로 설정했습니다. (명세서에 기인)
      위 점화식을 recursive 하게 구현, memoization 등을 이용하였습니다.
      시간복잡도는 간략하게 n^2의 복잡도를 n번 수행하기때문에 $O(n^3)$ 입니다.

   b. Subset sum

      점화식은 밑과 같이 구성했습니다.

$$D_{i,j} = \begin{cases} 1, & if\ j = 0\ or\ set[i] \\ D_{i-1,j}\ |\ D_{i-1,j-set[i]}, & otherwise \end{cases}$$

      주어지는 set 의 원소는 양수이기때문에, L 이 0 인 경우는 절대 찾을 수 없습니다. (empty set 인 경우는 count 하지 않기 때문에)
      L 이 0 이 아닌 경우, n*L size 의 bool 형 2d array 를 사용, 답을 도출했습니다.
      시간복잡도는 n*L size 의 2d array 를 전부 돌기때문에, $O(nL)$ 입니다.

2. Code
   a. Minimal Triangulation

      Cmp function to use quicksort

```
int cmp(struct I *a, struct I *b) { return a->i^b->i?a->i-b->i:a->j-b->j; }
```

      Dist function to calculate distance between two vertices

```
double dist(int a, int b)
{
    if(abs(a-b) == 1 || abs(a-b) == n-1) return 0;
    return sqrt((x[a]-x[b])*(x[a]-x[b])+(y[a]-y[b])*(y[a]-y[b]));
}
```

Struct to save indices, DP table, Points array

```
int n, pair[222][222], top;
double x[222], y[222], d[222][222];

struct I{
    int i, j;
}idx[222];
```

DP function (Recursive)

```
double tri(int f, int s)
{
    if(s < 4) return 0.0;
    if(d[f][s]) return d[f][s];
    // cutting & memoization

    int i;
    double ret = 1.7e307, a, b, c;
    for(i = 1; i < s-1; i++)
    {
        a = tri(f, i+1);
        b = tri(f+i, s-i);
        c = dist(f, f+i) + dist(f+i, f+s-1);
        if(ret > a+b+c)
        {
            ret = a+b+c;
            pair[f][s] = i;
        }
    }
    d[f][s] = ret;
    return ret;
}
```

Main Function

```c
int main()
{
    FILE *fp, *tcfp;
    int t, tc, n, i, j;
    double a;
    char in[22], out[22];

    fp = fopen("PT_test_command.txt", "r");
    fscanf(fp, "%d\n", &tc);
    for(t = 0; t < tc; t++)
    {
        fscanf(fp, "%s%s", in, out);
        // read input file name & output file name

        tcfp = fopen(in, "r");
        fscanf(tcfp, "%d", &n);
        for(i = 0; i < n; i++)
            fscanf(tcfp, "%lf%lf", x+i, y+i);
        fclose(tcfp);
        // get points

        top = 0;
        memset(d, 0.0, sizeof(d));
        memset(pair, 0, sizeof(pair));
        a = tri(0, n);
        find_pairs(0, n);
        // set dp table

        qsort(idx, top, sizeof(struct I), cmp);
        // sort indices

        tcfp = fopen(out, "w");
        fprintf(tcfp, "%lf\n", a);
        for(i = 0; i < top; i++)
            fprintf(tcfp, "%d %d\n", idx[i].i, idx[i].j);
        fclose(tcfp);
        // write the answer with given points
    }
    fclose(fp);
    return 0;
}
```

Find indices Fuction

```
void find_pairs(int f, int s)
{
     if(s < 4) return;
     int cur = pair[f][s];
     if(cur > 1)
     {
          idx[top].i = f;
          idx[top++].j = f+cur;
     }
     if(s-cur > 2)
     {
          idx[top].i = f+cur;
          idx[top++].j = f+s-1;
     }
     if(s < n)
     {
          idx[top].i = f;
          idx[top++].j = f+s-1;
     }
     find_pairs(f, cur+1);
     find_pairs(f+cur, s-cur);
}
```

b. Subset sum

(다음장에 있습니다)

Find indices iteration

```
if(L)
{
    fprintf(tcfp, "%d\n", d[n-1][L]);
    if(d[n-1][L])
    {
        for(i = n; i--; )
        {
            if(i && d[i-1][L]) continue;
            if(!i && !L) continue;
            L -= set[i];
            sel[idx++] = i;
        }
        fprintf(tcfp, "%d\n", idx);
        while( idx-- )
            fprintf(tcfp, "%d\n", sel[idx]);
    }
} else
    fprintf(tcfp, "0\n");
```

DP iteration

```
for(i = 0; i < n; i++)
    for(j = 0; j <= L; j++)
        d[i][j] = 0;
//initailize dp table

for(i = 0; i < n; i++)
{
    d[i][0] = d[i][set[i]] = 1;
    for(j = 1; j <= L; j++)
    {
        if(i) d[i][j] = d[i-1][j];
        if(i && j >= set[i]) d[i][j] |= d[i-1][j-set[i]];
    }
}
// set and fill n*L size dp table
```

3. 시간측정
    a. Minimal Triangulation

```
139:3_1 uppo97$ !.
./a.out
With Size 8, 0.010ms
With Size 19, 0.065ms
```

    b. Subset sum

```
139:3_2 uppo97$ !.
./a.out
With n is 9 and L is 0, 0.002ms
With n is 10 and L is 5842, 0.502ms
```

4. 실험 환경
    a. OS : OS X El Capitan Ver. 10.11.6
    b. CPU: 1.7 GHz Intel Core i7
    c. Ram: 8.00GB
    d. Compiler: gcc compiler Ver. 4.2.1 with -o optimization option