

论题 1-11 作业

姓名：陈劭源

学号：161240004

1 [DH] Problem 4.1

- (a) $S \leftarrow 0$;
for i going from 1 to N do the following:
 if $A[i, 1] > A[A[i, 2], 1]$ then $S \leftarrow S + A[i, 1]$;
output S .

- (b) Suppose the root of the binary tree is R .

$S \leftarrow 0$;
 $P \leftarrow R$;
 $N \leftarrow$ the content of the first offspring of R ;
if the content of $R >$ **get the salary of N th employee** then $S \leftarrow S +$ the content of R ;
while P has a second offspring do the following:
 $P \leftarrow$ the second offspring of P ;
 $N \leftarrow$ the content of the first offspring of S ;
 if the content of $P >$ **get the salary of N th employee** then $S \leftarrow S +$ the content of P ;
output S .

subroutine **get the salary of N th employee**

$T \leftarrow R$;
do the following $N - 1$ times:
 $T \leftarrow$ the second offspring of T ;
return the content of T ;

2 [DH] Problem 4.2

- (a) $S \leftarrow 0$;
call **add**($T, 0$);
output S .

subroutine **add**(P, x)

$S \leftarrow S + x$;
 $N \leftarrow 1$;

while P has an N th offspring do the following:
 call **add**(the N th offspring of P , $x + 1$);
 $N \leftarrow N + 1$;
return.

(b) $S \leftarrow 0$;
 call **count**(T , 0);
 output S ;

subroutine **count**(P , x)
 if $x = K$ then do the following:
 $S \leftarrow S + 1$;
 return;
 $N \leftarrow 1$;
 while P has an N th offspring do the following:
 call **count**(the N th offspring of P , $x + 1$);
 $N \leftarrow N + 1$;
 return.

(c) $R \leftarrow \text{false}$;
 call **check**(T , 0);
 output R .

subroutine **check**(P , x)
 if x is even then do the following:
 if P doesn't have a first offspring then do the following:
 $R \leftarrow \text{true}$;
 return;
 $N \leftarrow 1$;
 while P has an N th offspring do the following:
 call **check**(the N th offspring of P , $x + 1$);
 $N \leftarrow N + 1$;
 return.

3 [DH] Problem 4.8

Suppose that the maximal distance between any two points on a polygon occurs between M and N . First, regard N as an arbitrary fixed point, and consider point M .

Case 1: M is in the polygon. Extend NM cutting the polygon at E (Figure 2(a)). NP is longer than than NM .

Case 2: M is on one edge of the polygon, but M is not a vertex (Figure 2(b)). Let the edge where M is on be AB . At least one of $\angle NMA$ and $\angle NMB$ is not less than 90 degrees. Assume, WLOG, that $\angle NMA \geq 90^\circ$. By

the law of sines, we get $NA > NM$.

Now, we have proved that for arbitrary N , the length of NM is maximal when M is a vertex of the polygon. Consider point N , we can prove that the length of MN is maximal when N is a vertex of the polygon likewise (Figure 2(c)). Hence, the maximal distance between any two points on a polygon occurs between two of the vertices. \square

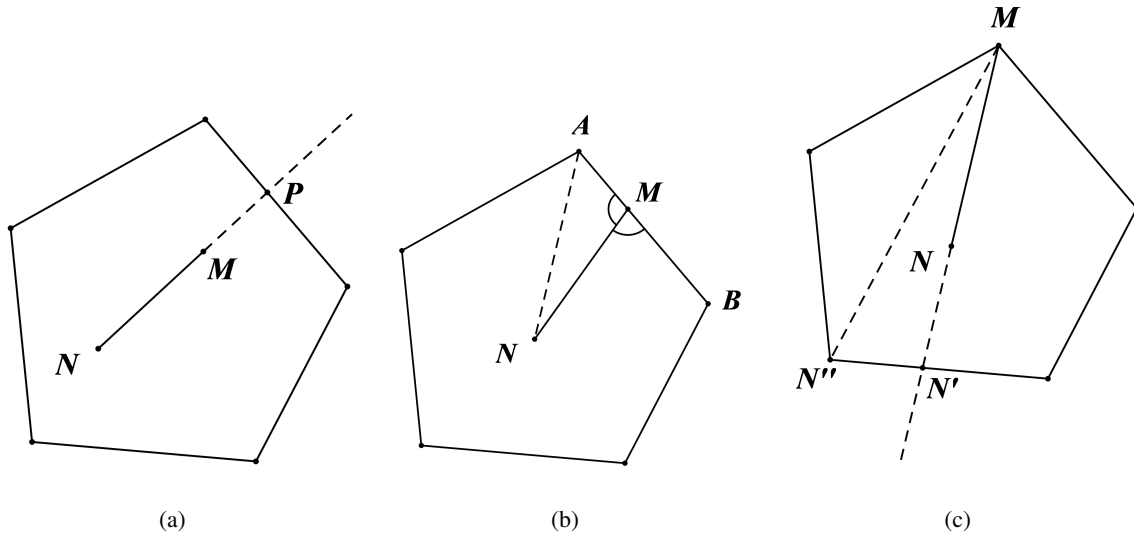


Figure 2: the distance of two points on a polygon

4 [DH] Problem 4.9

Language: C++

The first line of the input contains a positive integer n , giving the number of the vertices of the polygon. The following $n + 1$ lines of the input contains the coordinates of the vertices (in clockwise or counterclockwise order). The x-coordinate and y-coordinate are separated by a space.

The output of the program is a number, the maximal distance of two points on the polygon.

```
#include <iostream>
#include <cmath>
#include <algorithm>
using namespace std;

int n;
double x[1000], y[1000];

double dist(int i1, int i2)
{
    return hypot(x[i1 % n] - x[i2 % n], y[i1 % n] - y[i2 % n]);
}

int main()
```

```

{
    double ans = 0;
    int j, k;
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> x[i] >> y[i];
    j = n;
    k = n + 1;
    while (k < 2*n)
    {
        while (!(dist(j, k) > dist(j, k - 1) && dist(j, k) > dist(j, k + 1)))
            k++;
        ans = max(ans, dist(j, k));
        j++;
    }
    cout << ans << endl;
    return 0;
}

```

5 [DH] Problem 4.11

Suppose the vector is named V .

- (a) $M_1 \leftarrow$ **find maximum of first N elements**;

$I \leftarrow 1$;

do the following while $V[I] \neq M_1$:

$I \leftarrow I + 1$;

for I going from I to $N - 1$ do the following:

$V[I] \leftarrow V[I + 1]$;

$M_2 \leftarrow$ **find maximum of first $N - 1$ elements**;

output M_1, M_2 .

subroutine **find maximum of first n elements**

$A \leftarrow V[1]$;

for i going from 2 to n do the following:

if $V[i] > A$ then $A \leftarrow V[i]$;

return A .

- (b) $M_1 \leftarrow$ **find maximum from 1th to N th element**;

$I \leftarrow 1$;

do the following while $V[I] \neq M_1$:

$I \leftarrow I + 1;$
 for I going from I to $N - 1$ do the following:
 $V[I] \leftarrow V[I + 1];$
 $M_2 \leftarrow$ **find maximum from 1th to $(N - 1)$ th element**;
 output M_1, M_2 .

subroutine **find maximum from m th to n th element**;
 if $m = n$ then return $V[m];$
 $p \leftarrow \lfloor (m + n) / 2 \rfloor;$
 $T_1 \leftarrow$ **find maximum from m th to p th element**;
 $T_2 \leftarrow$ **find maximum from $(p + 1)$ th to n th element**;
 if $T_1 > T_2$ then return $T_1;$
 otherwise return T_2 .

6 [DH] Problem 4.12

Suppose there are M nodes and N edges in the graph, the nodes are numbered from 1 to M and the edges are stored in vector V . Every edge T support three operations: get the number of the first node it connects (T .first), get the number of the second node it connects (T .second) and get the length of the node (T .length). Let U be an empty vector of integers. The output is the edges forming the minimal spanning tree.

call **initialize**;
 call **quicksort from 1 to N** ;
 $m \leftarrow 0;$
 $i \leftarrow 1;$
 while $m < M - 1$ do the following:
 if **find** $V[i]$.first \neq **find** $V[i]$.second then do the following:
 call **union** $V[i]$.first **and** y .first;
 output $V[i];$
 $m \leftarrow m + 1;$
 $i \leftarrow i + 1$.

subroutine **initialize**
 for i going from 1 to N do the following:
 $U[i] = i;$

subroutine **find** x
 if $U[x] = x$ then return $x;$
 $t \leftarrow$ **find** $U[x];$
 $U[x] \leftarrow t;$
 return t .

subroutine **union** x **and** y

$p \leftarrow \text{find } x;$

$q \leftarrow \text{find } y;$

$U[p] \leftarrow q.$

subroutine **quicksort** from a to b

if $a \geq b$ then return;

$p \leftarrow \text{partition from } a \text{ to } b;$

call **quicksort** from a to $p - 1$;

call **quicksort** from $p + 1$ to b .

subroutine **partition** from a to b

call **swap** $\lfloor (a + b)/2 \rfloor$ **and** b ;

$L \leftarrow a$;

for i going from a to $b - 1$ do the following:

if $V[i].\text{length} < V[b].\text{length}$ do the following:

call **swap** i **and** L ;

$L \leftarrow L + 1$;

call **swap** b **and** L ;

return L .

subroutine **swap** a **and** b

$t \leftarrow V[a];$

$V[a] \leftarrow V[b];$

$V[b] \leftarrow t;$

return.

7 [DH] Problem 4.13

(a) Let R be an empty vector of integers, S be an empty two-dimensional array of integers.

for i going from 0 to C do the following:

$R[i] \leftarrow 0$;

for j going from 1 to N do the following:

$S[i][j] = 0$;

for i going from 1 to N do the following:

for j going from 1 to $Q[i]$ do the following:

for k going down from C to $W[i]$ do the following:

if $R[j - W[i]] + P[i] > R[j]$ do the following:

$R[j] \leftarrow R[j - W[i]] + P[i];$

for l going from 1 to i do the following:

```

         $S[j][l] \leftarrow S[j - W[i]][l];$ 
         $S[j][i] \leftarrow S[j][i] + 1;$ 
    output  $S[C]$ .

```

(b) The output is [0, 1, 3, 2, 1]. The total profit of the knapsack is 194.

8 [DH] Problem 4.14

(a) Let S be an empty vector of real numbers.

```

while  $C \neq 0$  do the following:
     $t \leftarrow$  find best material;
    if  $W[t] \times Q[t] < C$  then do the following:
         $C \leftarrow C - W[t] \times Q[t];$ 
         $Q[t] \leftarrow 0;$ 
         $S[t] \leftarrow Q[t];$ 
    otherwise do the following:
         $Q[t] \leftarrow Q[t] - C/W[t];$ 
         $S[t] \leftarrow C/W[t];$ 
         $C \leftarrow 0;$ 
output  $S$ .

```

subroutine **find best material**

```

     $i \leftarrow 1;$ 
    while  $Q[i] = 0$  do the following:
         $i \leftarrow i + 1;$ 
     $t \leftarrow i;$ 
    for  $i$  going from  $i + 1$  to  $N$  do the following:
        if  $Q[i] > 0$  and  $P[i]/W[i] > P[t]/W[t]$  then  $t \leftarrow i;$ 
    return  $t$ .

```

(b) The output is [0, 1, 1.8, 5, 1]. The total profit of the knapsack is 200.