Report Term Project

Algorithm Design (CS3201)

Section 541

Semester 1/2016

By Chatchawan yoojuie 5715298

Anniversary Party – Question Code: 1039

Background

The president of the Ural State University is going to make an 80'th Anniversary party. The university has a hierarchical structure of employees; that is, the supervisor relation forms a tree rooted at the president. Employees are numbered by integer numbers in a range from 1 to *N*, The personnel office has ranked each employee with a conviviality rating. In order to make the party fun for all attendees, the president does not want both an employee and his or her immediate supervisor to attend.

Problem

Your task is to make up a guest list with the maximal conviviality rating of the guests.

Input

The first line of the input contains a number *N*. $1 \le N \le 6000$. Each of the subsequent *N* lines contains the conviviality rating of the corresponding employee. Conviviality rating is an integer number in a range from –128 to 127. After that the supervisor relation tree goes. Each line of the tree specification has the form

<L> <K>

which means that the *K*-th employee is an immediate supervisor of *L*-th employee. Input is ended with the line

0 0

Output

The output should contain the maximal total rating of the guests.

Sample

Input	Output
7 1 1 1 1 1 1 1 3 2 3 6 4 7 4 4 5 3 5 0 0	5

Problem Author: Marat Bakirov
Problem Source: Ural State University Internal Contest October'2000 Students Session
Difficulty: 380
Time limit: 0.5 second
Memory limit: 8 MB

Solution

The class is created in-order to store the information of the node.

```
class employee:
	def __init__(self, n, c):
		self.n = n
		self.inFun = c
		self.exFun = 0
		self.subtree = 0
		self.boss = None
		self.member = []
```

For the input part, all the N lines contains the conviviality rating will be store in array. And the relation between nodes are update in the loop.

```
from employee import employee

N = input()
E = []

for i in range(N):
	x = input()
	E.append(employee(i + 1, x))

while 1:
	x = raw_input().split()
	l = int(x[0])
	k = int(x[1])
	if l == 0 and k == 0:
		break
	else:
		E[l - 1].boss = k - 1  # index start from 0
		E[k - 1].member.append(E[l - 1])
```

To solve this problem, depth-first search is the solution. So, the stack is required to store the node and pop it out once all subtree has been visited.

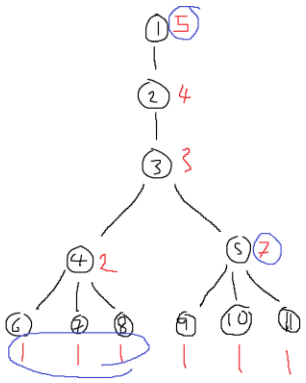
```
S = []  # stack
mx = 0

for i in range(N):
	if E[i].boss is None:
		S.append(E[i])

while S:
	v = S[len(S) - 1]
	if v.subtree < len(v.member):
		S.append(v.member[v.subtree])  # push next subtree in the stack
		v.subtree += 1
	else:
		for x in v.member:
			v.exFun += max(x.exFun, x.inFun)
			v.inFun += x.exFun
		if v.boss is None:
			mx += max(v.exFun, v.inFun)
		S.pop()
print mx
```

Time complexity

Depth-first search is used to solve this problem. It is starts at the root of the tree and explores as far as possible along each branch before backtracking. And calculate the total maximum conviviality for including and excluding the root node in each subtree that has been visited.

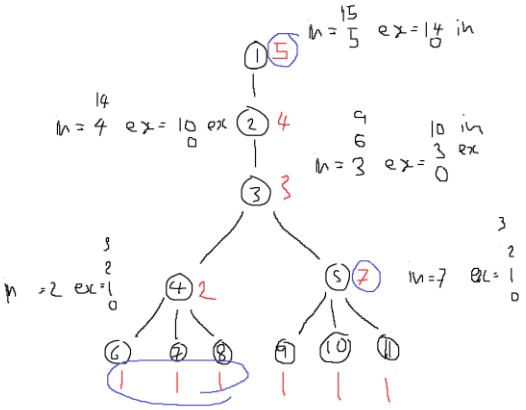
It is linear time algorithm where O(V+E) or O(E) if O(V) is constant.

ID	Date	Author	Problem	Language	Judgement result	Test #	Execution time	Memory used
7151770	22:55:38 30 Nov 2016	Cwan	1039. Anniversary Party	Python 2.7	Accepted		0.171	4 452 KB



```
11
5
4
3
2
7
1
1
1
1
1
1
6 4
7 4
8 4
9 5
10 5
11 5
4 3
5 3
3 2
2 1
0 0
```

Answer:15



```
11
5
4
3
2
7
1
1
1
1
1
1
6 4
7 4
8 4
9 5
10 5
11 5
4 3
5 3
3 2
2 1
0 0
```

Answer:15