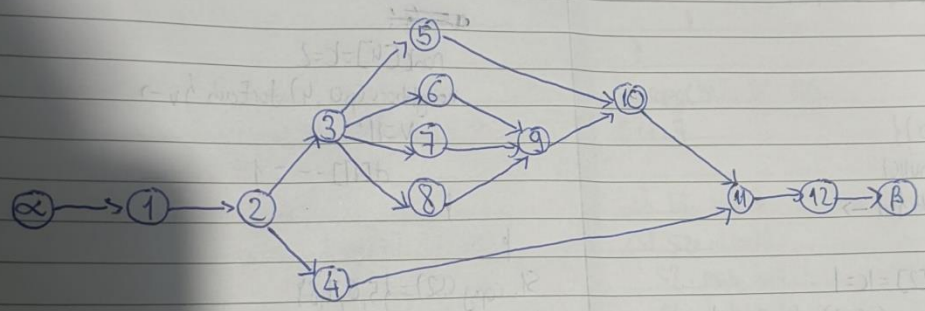
Trần Nguyễn Nhật Huy
B2113333
56.

1. Mô hình hoá bài toán
- Công việc ⇒ đỉnh.
- Sự phụ thuộc giữa 2 công việc ⇒ Cung
- Thời gian thực hiện ⇒ trọng số

- Thêm 2 đỉnh α, β tương ứng với 2 công việc giả
- Thêm cung nối α với các đỉnh có bậc vào bằng 0
- Thêm cung nối các đỉnh có bậc ra bằng 0 với β

2.




3.

```
int ranking[MAX_VERTICES]

void ranking (Graph *pG){
    Int d[MAX_VERTICES]
    d[1] = 0,
    d[2] = d[3] = d[4] = 1
    d[5] = d[6] = d[7] = d[8] = 1
    d[9] = 3
    d[10] = d[11] = 2
    d[12] = 12
    List S1, S2
    S1 = d.withIndex().filter{u -> d[u] == 0}.map{u -> u.index}
    S1 = {0}
```

int k=0

@ Lần lặp 1

(S1. size = 1>0){

S2. make_null()

S1. forEach { u ->

rank[u=1] = k=0

neighbors(pG, 1). forEach { v ->

v=2:

d[2]-- =0

S2. push_back(2), S2={2}

}

}

S1. copy(S2), S1={2}

k++ = 1

}

@ Lần lặp 2:

(S1. size = 1>0){

S2. make_null()

S1. forEach { u ->

u=2:

rank[2] = k=1

neighbors(pG, 2). forEach { v ->

v=3, 4:

d[3, 4]-- =0

S2. push_back(3,4), S2={3,4}

}

}

S1. copy(S2), S1={3,4}

k++ = 2

}

@ Lần lặp 3:

(S1. size = 2>0){

S2. make_null()

S1. forEach { u ->

u=3:

rank[3] = k=2

neighbors(pG, 3). forEach { v ->

v=5,6,7,8

d[5,6,7,8]-- =0

S2. push_back(5,6,7,8), S2={5,6,7,8}

}

}

~~S1. copy(S2), S1={5,6,7,8}~~

~~k++ = 3~~

~~@ Lần lặp 3:~~

~~(S1. size = 4>0){~~

~~S2. make_null()~~

~~S1. forEach { u ->~~

~~u=4:~~

rank[4] = k=2

neighbors(pG, 4). forEach { v ->

v=11:

d[11]-- = 1

}

}

S1. copy(S2) = {5,6,7,8}

k++ = 3

}

@ Lần lặp 4:

(S1. size = 4>0){

S2. make_null()

S1. forEach { u ->

u=5:

rank[5] = k=3

neighbors(pG, 5). forEach { v ->

v=10:

d[10]-- = 1

}

u=6,7,8:

rank[6,7,8] = k=3

neighbors(pG, 6,7,8). forEach { v ->

=15,6,7,8}

v=9
d[9]--=2
d[8]--=1
d[9]--=0
S2.push_back(9), S2={9}
}
}
S1.copy(S2)={9}
k++=4
}
@ Lần lặp 5
(S1.size=1>0){
S2.make_null()
S1.forEach{u->
u=9:
rank u=k=4
neighbors(pG,9).forEach{v->
v=10:
d[10]--=0
S2.push_back(10), S2={10}
}
}
S1.copy(S2)={10}
k++=5
}
@ Lần lặp thứ 6
(S1.size=1>0){
S2.forEach{u->
u=10:
rank[10]=k=5
neighbors(pG,10).forEach{v->
v=11:
d[11]--=0
S2.push_back(11), S2={11}
}
}

S1.copy(S2), S1={11}
k++=6
}
@ Lần lặp thứ 7
(S1.size=1>0){
S2.make_null()
S1.forEach{u->
u=11:
rank[11]=k=6
neighbors(pG,11).forEach{v->
v=12:
d[12]--=0
S2.push_back(12), S2={12}
}
}
S1.copy(S2), S1={12}
k++=7
}
@ Lần lặp thứ 7
(S1.size=1>0){
S2.make_null()
S1.forEach{u->
u=12:
rank[12]=k=7
neighbors(pG,12).forEach{v->
v=null
}
}
S1.copy(S2), S1={}
k++=8
}
@ Lần lặp thứ 8
(S1.size=0){
break
}

rank[1] = 0
rank[2] = 1
rank[3] = rank[4] = 2
rank[5] = rank[6] = rank[7] = rank[8] = 3
rank[9] = 4
rank[10] = 5
rank[11] = 6
rank[12] = 7