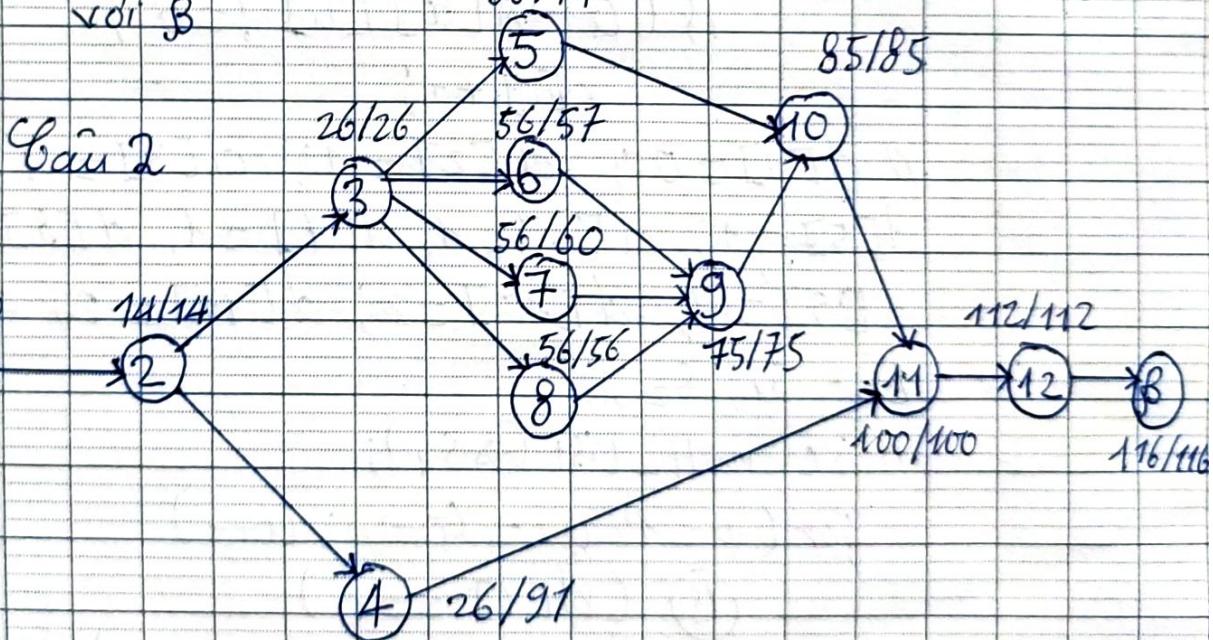


Câu 1) Mô hình hóa bài toán bằng lý thuyết đồ thị

- * Mũ công việc \Rightarrow Đỉnh
- * Sư phụ thuộc giữa 2 công việc \Rightarrow cung
- * Thời gian hoàn thành \Leftrightarrow u (drew)
- * Của đỉnh u \Rightarrow trọng số
- * Thêm 2 đỉnh a, b (tương ứng với 2 công việc giờ)
- * Thêm cung nối a nối với các đỉnh có bắt vào bằng 0
- * Thêm cung nối các đỉnh có bắt ra bằng 0

Câu 2



Góp 3: Chạy thuật toán giải thuật xếp hàng có
thứ:

```

(X) int rank [MAX_VERTICES];
void ranking (Graph *G) {
    int d[MAX_VERTICES];
    int x, u;
    for (u=0; u<=G->n; u++)
        d[u]=0;
    // Tạo các đỉnh đều bằng 0
    for (x=1; x<=G->n; x++)
        for (u=1; u<=G->n; u++)
            if (G->A[x][u] != 0)
                d[u]++;
    // d[1]=0; d[2]=1; d[3]=1; d[4]=1;
    // d[5]=1; d[6]=1; d[7]=1; d[8]=1;
    // d[9]=3; d[10]=2; d[11]=2; d[12]=1;
    List S1, S2;
    make_null_list(&S1);
    for (u=1; u<=G->n; u++)
        if (d[u]==0)
            push_back(&S1, 1),
    int k=1, i;
    ① while (S1.size>0) {
        make_null_list(&S2);
        for (int i=1; i<=S1.size; i++)
            int u=element_at(&S1, 1);

```

```

rank[1] = 1;
int v,
for (int v=1; v<=G->n; v++) {
    if (G->A[1][2] != 0) {
        d[2]--; // d[2]=0;
        if (d[2] == 0)
            push_back(&s2, 2);
    }
}
copy_list(&s1, &s2); // s1{2};
K = 2;
*  

② while (s1.size > 0) { // s1{2};
    make null list (&s2);
    for (i=1; i<=s1.size; i++) {
        int u = element_at(&s1, 1);
        rank[2] = 2;
        for (int v=1; v<=G->n; v++)
            if (G->A[2][3] != 0)
                d[3]--; // d[3]=0;
            if (d[3] == 0)
                push_back(&s2, 3);
        if (G->A[2][4] != 0)
            d[4]--; // d[4]=0;
            if (d[4] == 0)
                push_back(&s2, 4);
    }
}

```

copy_list(>ss₁, >ss₂); // s₁ {3, 4};
 k = 3;

(3)

```

while (s1.size > 0) {
  make_null_list(>ss2);
  for (i = 1; i <= 2; i++) {
    int u = element_at(>s1, i);
    rank[E3] = 3;
    for (int v = 1; v <= G->n; v++)
      if (G->A[E3][E5] != 0) {
        d[E5]--;
        if (d[E5] == 0)
          push_back(>ss2, v);
      }
    if (G->A[E3][E6] != 0) {
      d[E6]--;
      if (d[E6] == 0)
        push_back(>ss2, 6);
    }
    if (G->A[E3][E7] != 0) {
      d[E7]--;
      if (d[E7] == 0)
        push_back(>ss2, 7);
    }
    if (G->A[E3][E8] != 0) {
      d[E8]--;
      if (d[E8] == 0)
        push_back(>ss2, 8);
    }
  }
  copy_list(>ss1, >ss2);
}
  
```

k = -

int u = element_at(&S1, 2);
 rank[u] = 3;
 for (int v = 1; v <= G->n; v++) {
 if (G->A[4][v] == 0)
 d[v]--; d[1] = 1;
 }
 copy_list(&S1, &S2); // S1 {5, 6, 78 }.
 k = 4;

10: (4) while (S1.size > 0) {
 make_null_list(&S2);
 for (int i = 1; i <= S1.size; i++) {
 int u = element_at(&S1, 1);
 rank[S] = u;
 for (v = 1; v <= G->n; v++) {
 if (G->E[S][v] == 0)
 d[v]--; // d[1] = 0;
 }
 int u = element_at(&S1, 2);
 rank[6] = u;
 for (v = 1; v <= G->n; v++) {
 if (G->E[6][v] == 0)
 d[v]--; // d[9] = 2;
 }
 int u = element_at(&S1, 3);
 rank[7] = u;
 for (v = 1; v <= G->n; v++) {
 if (G->E[7][v] == 0)
 }

1. $d[9]--;$ // $d[9] = 1;$

int u = element_at(&S1, 4);
 $\text{rank}[8] = u;$

for (int v = 1; v <= C ->n; v++) {

(C->[8][9])! = 0;

$d[9]--;$

(d[9] == 0)

push_back(&S2, 9);

}

copy_list(&S1, &S2); // S1 \{ 9 \};

k = 5;

}

⑤ while (S1.size > 0) {

make null_list(&S2);

for (i = 1; i < S1.size; i++) {

int u = element_at(&S1, i);

rank[9] = i;

for (int v = 1; v <= C->n; v++)

(C->[9][10])! = 0;

$d[10]--;$

(d[10] == 0)

push_back(&S2, 10);

}

copy_list(&S1, &S2); // S1 \{ 10 \};

k = 6;

25:

3

⑥ while ($S_1.size > 0$) {
 make null list ($\&S_2$);
 for ($i = 1$; $i < S_1.size$; $i++$)
 int u = element at ($\&S_1, i$);
 rank [10] = 6;
 for (int v = 1, $v <= G - n$; $v++$) {
 ($G \rightarrow A[10][v] = 0$)
 $c[11]--$;
 ($c[11] = 0$)
 push back ($\&S_2, 11$);
 }
 copy list ($\&S_1, \&S_2$);
 k = 7;
 }
 }

⑦ while ($S_1.size > 0$) {
 make null list ($\&S_2$);
 for ($i = 1$; $i < S_1.size$; $i++$) {
 int u = element at ($\&S_1, i$);
 rank [11] = 7;
 for (int v = 1, $v <= G - n$; $v++$) {
 ($G \rightarrow A[11][v] = 0$)
 $d[12]--$;
 ($d[12] = 0$)
 push back ($\&S_2, 12$);
 }
 copy list ($\&S_1, \&S_2$);
 k = 8;

(8)

```
while (S1.size > 0) {
    make-null-list (DS2);
    for (int i = 1; i < S1.size; i++) {
        int u = element-at (&S1, 1);
        rank[12] = 8;
        for (int v = 1; v <= (o - 1); v++) { } }
```

10 = 9;

{}

10.

15.

20.

25.