Make Span Schedule

A project has n tasks 1,. . ., n. Task i has duration d(i) to be completed (i=1,. . ., n). There are precedence constraints between tasks represented by a set Q of pairs: for each (i,j) in Q, task j cannot be started before the completion of task i. Compute the earliest completion time of the project.

Input

- Line 1: contains n and m (1 <= n <= 10^4, 1 <= m <= 200000)
- Line 2: contains d(1),. . ., d(n) (1 <= d(i) <= 1000)
- Line i+3 (i=1,. . ., m) : contains i and j : task j cannot be started to execute before the completion of task i

Output

- Write the earliest completion time of the project.

Make Span Schedule

Input

9 13

5 3 1 2 6 4 3 1 4

1 3

1 5

1 6

2 1

2 3

3 5

4 1

4 2

4 6

5 8

7 9

9 5

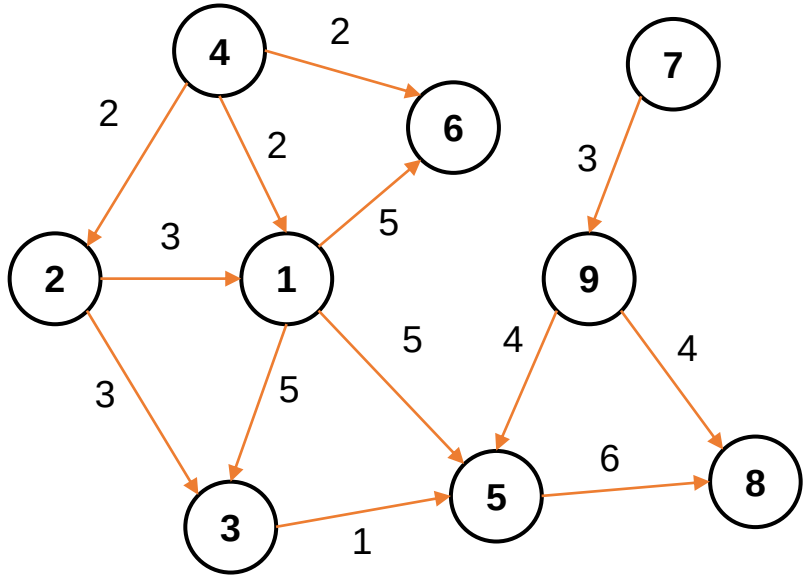
9 8

Output

18

Make Span Schedule

- Algorithm
  - L is the TOPO list of nodes of G
  - F[u]: earlest time point the task u can start
  - Explore L from left to right, for each node u:
    - makespan = max(makespan, F[u] + d[u])
    - For each arc (u,v), update F[v] = max(F[v], F[u] + d[u])



Implementation – Make Span Schedule

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6;
struct Arc{
    int v;
    int w;
    Arc(int _v, int _w): v(_v), w(_w){}
};
int n,m;
int duration[N];
vector<Arc> A[N];// A[v] set of outgoing arc of v
int d[N];// incoming degree
vector<int> L;
int F[N];// F[v] earliest possible starting time-point
int ans;
```

Implementation – Make Span Schedule

```
void input(){
    memset(d,0,sizeof d);
    cin >>n >> m;
    for(int i = 1; i <= n; i++)
        cin >> duration[i];
    for(int k = 1; k <= m; k++){
        int u,v;
        cin >> u >> v;
        A[u].push_back(Arc(v,duration[u]));
        d[v]++;
    }
}
```

Implementation – Make Span Schedule

```
void topoSort(){
    queue<int> Q;
    for(int v = 1; v <= n; v++) if(d[v] == 0)
        Q.push(v);
    while(!Q.empty()){
        int x = Q.front(); Q.pop();
        L.push_back(x);
        for(int i = 0; i < A[x].size(); i++){
            int y = A[x][i].v;
            int w = A[x][i].w;
            d[y] -= 1;
            if(d[y] == 0) Q.push(y);
        }
    }
}
```

Implementation – Make Span Schedule

```
void solve(){
    memset(F,0,sizeof F);
    ans = 0;
    for(int i = 0; i < L.size(); i++){
        int u = L[i];
        ans = max(ans,F[u] + duration[u]);
        for(int j = 0; j < A[u].size(); j++){
            int v = A[u][j].v;
            int w = A[u][j].w;
            F[v]  = max(F[v],F[u] + w);
        }
    }
    cout << ans << endl;
}
int main(){
    input();
    topoSort();
    solve();
}
```