HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Applied Algorithm Lab

Make span schedule

ONE LOVE. ONE FUTURE.

Make span schedule

- Make a schedule for a project with many tasks.
- A project has n tasks 1,..., n:
  - Task i has duration d(i) to be completed
  - Precedence constraints Q: for each (i,j) in Q, task j cannot be started before the completion of task i.
- **Objective:** Arrange task the project to complete as soon as possible.
- **Input:** n, |Q|, d(1),..., d(n), the set Q
- **Output:** The earliest completion time of the project.

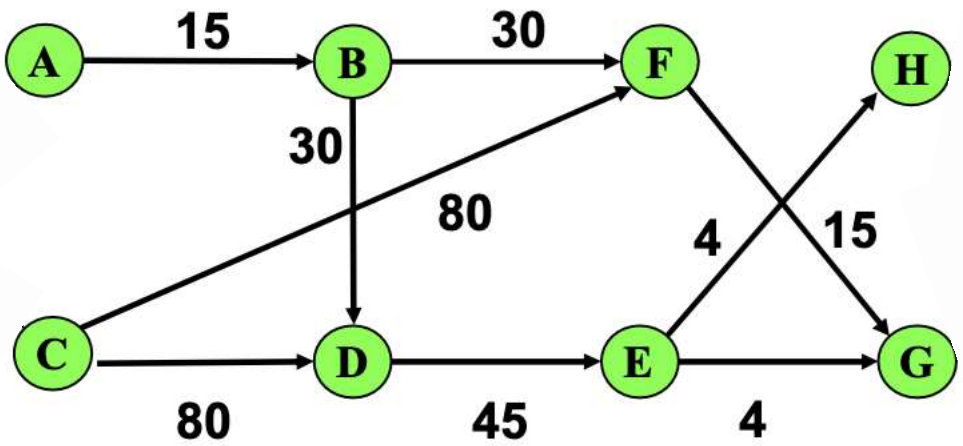
Make span schedule

- Example

Input	Output
8 9 15 30 80 45 4 15 15 19 1 2 2 4 3 4 4 5 2 6 3 6 5 7 6 7 5 8	148

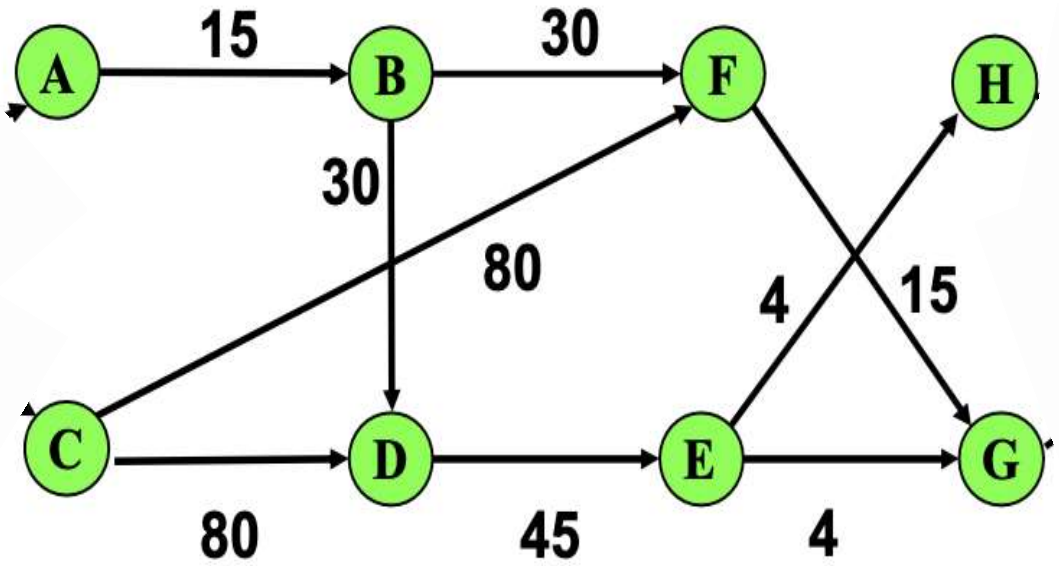
Make span schedule

- Idea to solve: Formulate the problem using directed graph
  - A graph with n nodes as n tasks
  - If (i,j) ∈ Q, we draw an edge (i, j) on graph
    - weight of edge (i,j) is t[i]
  - The node having in-degree 0 : we can perform the corresponding task since t = 0



Make span schedule

- Idea to solve: BFS
  - The node having in-degree 0 -> not affected by any other, but may affect some other nodes
  - We use BFS to traverse, from a node y having in-degree 0
    - Use auxiliary variable dist[x]: the first starting time of x
  - After BFS all node: retrieve the node with maximal dist[]



Make span schedule - Implementation

```cpp
#include <bits/stdc++.h>
using namespace std;

#define N 10001
int n, m, in_deg[N];
vector<int> Adj[N];
int t[N], firstTime[N], finishTime[N];
int min_finishTime = 0;

void input() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &t[i]);
        firstTime[i] = 0;
    }
    for (int i = 1; i <= m; i++) {
        int u, v;
        scanf("%d%d", &u, &v);
        Adj[u].push_back(v);
        in_deg[v]++;
    }
}
```

Make span schedule - Implementation

```cpp
int main() {
    input();
    for (int i = 1; i <= n; i++) {
        if (in_deg[i] == 0) { // BFS từ i
            firstTime[i] = 0;
            queue<int> q;
            q.push(i);
            while (!q.empty()) {
                int u = q.front();
                q.pop();
                for (int v : Adj[u]) {
                    if (firstTime[v] < firstTime[u] + t[u]) {
                        firstTime[v] = firstTime[u] + t[u];
                        finishTime[v] = firstTime[v] + t[v];
                        q.push(v);
                    }
                }
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        min_finishTime = max(min_finishTime, finishTime[i]);
    }
    cout << min_finishTime << endl;
    return 0;
}
```

THANK YOU !