

The background of the entire image is a dark blue field filled with a pattern of red dots. These dots are arranged in a way that they form a large, faint, stylized circular shape, reminiscent of a DNA helix or a molecular structure, with the density of the dots varying to create a sense of depth and movement.

# 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  $1, \dots, n$ :
  - 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:**  $n$ ,  $|Q|$ ,  $d(1), \dots, d(n)$ , the set  $Q$
- **Output:** The earliest completion time of the project.

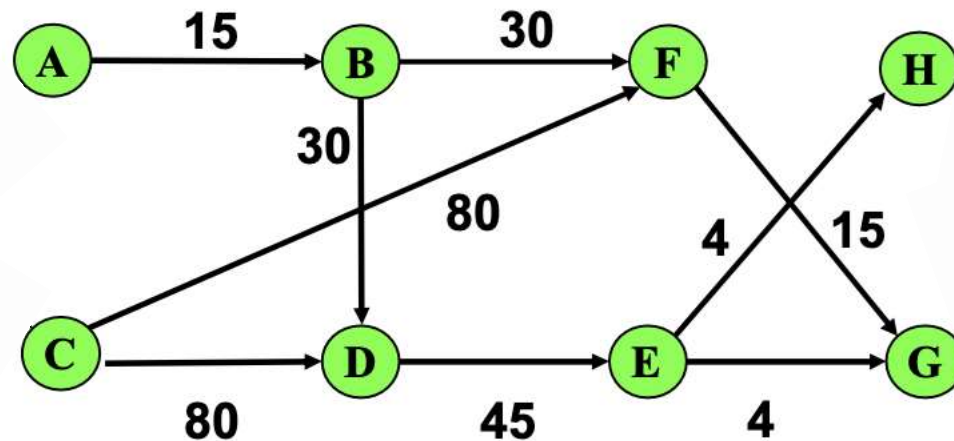
# Make span schedule

- Example

Input	Output
8 9	1 4 8
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	

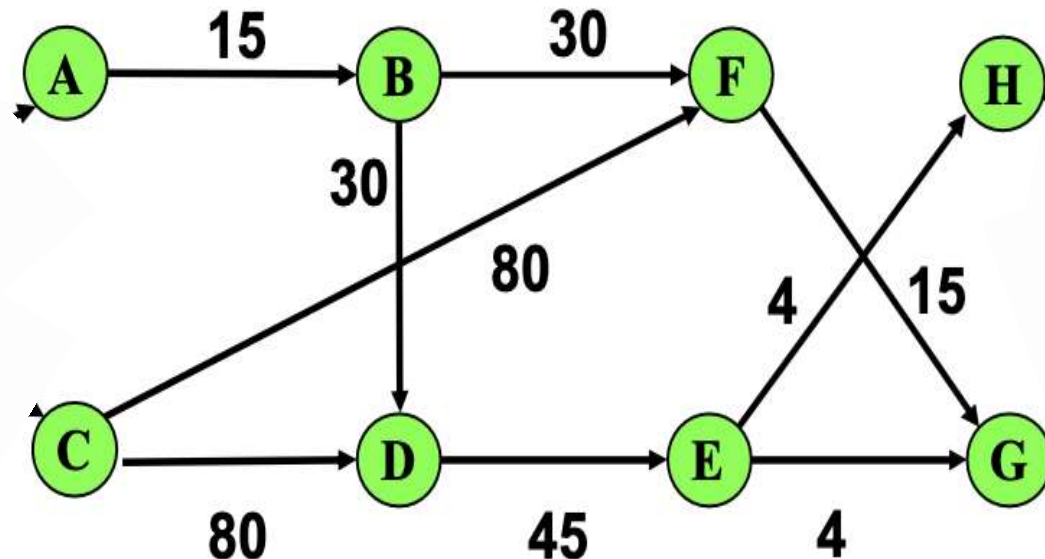
# Make span schedule

- Idea to solve: Formulate the problem using directed graph
  - A graph with  $n$  nodes as  $n$  tasks
  - If  $(i,j) \in Q$ , 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  $\text{dist}[x]$ : the first starting time of  $x$
  - After BFS all node: retrieve the node with maximal  $\text{dist}[]$





# Make span schedule - Implementation

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define N 10001
5  int n, m, in_deg[N];
6  vector<int> Adj[N];
7  int t[N], firstTime[N], finishTime[N];
8  int min_finishTime = 0;
9
10 void input() {
11     scanf("%d%d", &n, &m);
12     for (int i = 1; i <= n; i++) {
13         scanf("%d", &t[i]);
14         firstTime[i] = 0;
15     }
16     for (int i = 1; i <= m; i++) {
17         int u, v;
18         scanf("%d%d", &u, &v);
19         Adj[u].push_back(v);
20         in_deg[v]++;
21     }
22 }
```

# Make span schedule - Implementation

```
24 int main() {
25     input();
26     for (int i = 1; i <= n; i++) {
27         if (in_deg[i] == 0) { // BFS từ i
28             firstTime[i] = 0;
29             queue<int> q;
30             q.push(i);
31             while (!q.empty()) {
32                 int u = q.front();
33                 q.pop();
34                 for (int v : Adj[u]) {
35                     if (firstTime[v] < firstTime[u] + t[u]) {
36                         firstTime[v] = firstTime[u] + t[u];
37                         finishTime[v] = firstTime[v] + t[v];
38                         q.push(v);
39                     }
40                 }
41             }
42         }
43     }
44     for (int i = 1; i <= n; i++) {
45         min_finishTime = max(min_finishTime, finishTime[i]);
46     }
47     cout << min_finishTime << endl;
48     return 0;
49 }
```



A large graphic on the left side of the slide. It features a dark blue background with a circular pattern of red dots of varying sizes, creating a sense of depth and movement. The word "HUST" is centered within this pattern in a white, bold, sans-serif font.

**HUST**

**THANK YOU !**