Backtracking: Đệ quy quay lui → giải các BT liệt kê / Tối ưu tổ hợp (vét cạn)

Sơ đồ chung

liệt kê các giá trị cho các biến quyết định $\boxed{x_1, x_2, \ldots x_n}$ thỏa mãn các đk cho trước hoặc (đồng khi) tối ưu 1 hàm mục tiêu

Xét lần lượt các biến (trái → phải) $x_1, x_2 \ldots x_k, x_{k+1}, \ldots, x_n$

mỗi biến → xét ~~tất~~ cả các g/trị có thể assign cho biến.

```
Try(k) // thử g/trị cho x_k
[ for v ∈ candidate(x_k) do
    if check(v, k) then
    [ x_k = v; // decision;
      [update D];
      if k = n then solution()
      else Try(k+1);
      [recover D]; // undo decision
    ]
]
```

Xét VD: liệt kê tất cả $(X_1, X_2, \ldots, X_n)$ sao cho $X_1 + X_2 + \ldots + X_{k-1} + \boxed{X_k} + X_{k+1} + \ldots + X_n = M$

$x_1, x_2 \ldots x_{k-1}$ đã có giá trị

(n, M cho trước, $X_1, X_2 \ldots X_n$ nguyên dương)

$n=3, M=5$

$1+1+3=5$

$1+2+2=5$

$1+3+1=5$

$2+1+2=5$

$2+2+1=5$

$3+1+1=5$

$1 \leq x_k \leq M-(n-1)$

```
check(v, k)
    T = X_1 + ... + X_{k-1} + v;
    if k < n then
        return T < M
    else
        return T = M
```

```
Try(k)
    for v = 1 → M-(n-1) do
        if check(v, k) then
            x_k = v;
            if k = n then Solution()
            else Try(k+1)
```

TSP: Bài toán người du lịch

solution model: $(x_1, x_2, \dots, x_n)$

Tour $\underbrace{x_1 \to x_2}\underbrace{\to x_3} \to \dots \to \underbrace{x_n \to x_1}$

$1 = x_1$

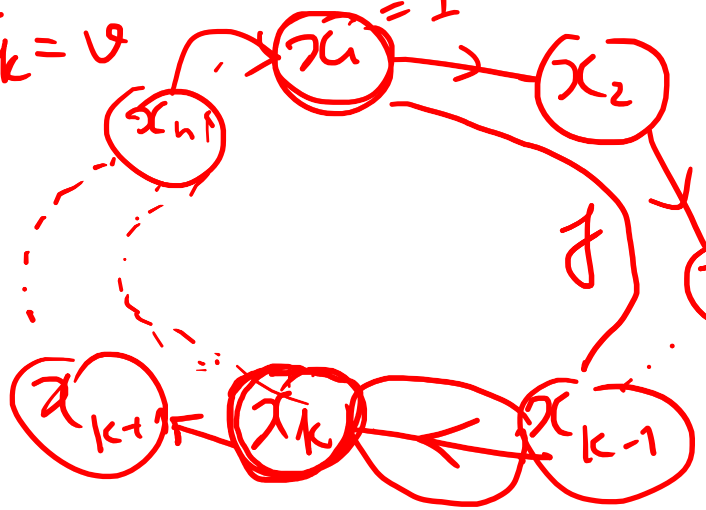
Độ dài lộ trình $\boxed{f} = c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_n, x_1)$

$x_i \in \{1, \dots, n\}$, không lặp giá trị

→ mảng đánh dấu

visited[v] = True nếu v đã xuất hiện

$x_k = v$



k-1

còn lại n-k+1 chặng

ký hiệu Cmin là chặng ngắn nhất

→ phần còn lại $\geq C_{min}(n-k+1)$

if $\boxed{f + C_{min}(n-k+1)} \geq f_{min}$ → đi tiếp

```
Try(k)
  for v = 1 → n do          Branch and Bound
    if check(v, k) then
      x_k = v;
      f = f + C(x_{k-1}, x_k);
      visited[v] = True;
      if k = n then solution()
      else if f + Cmin(n-k+1) < fmin
        Try(k+1)
      f = f - C(x_{k-1}, x_k)
      visited[v] = false;

check(v, k)
  return visited[v] = false
```

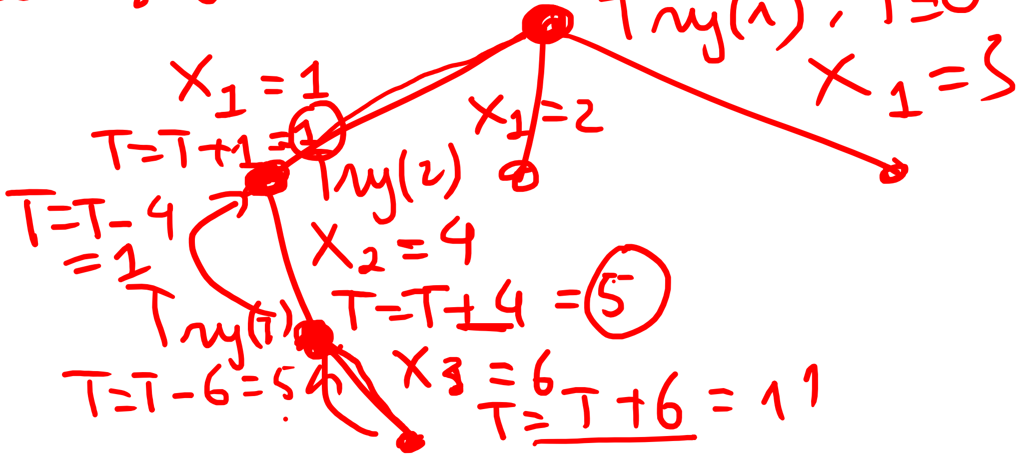
Duy trì biến tổng thể T
là tổng các biến đã được gán giá trị

$$\underbrace{x_1+x_2+\dots+x_{k-1}}_{T} + \underset{\uparrow v}{x_k} + \underbrace{x_{k+1}+\dots+x_n}_{\text{unknown} \,\geq\, n-k} = M$$

$$1 \le x_k \le M-T-(n-k) \le M-(n-1)$$

Decision $x_k = v \to$ update $T = T + x_k$

undo decision $\to$ recover $T = T - x_k$

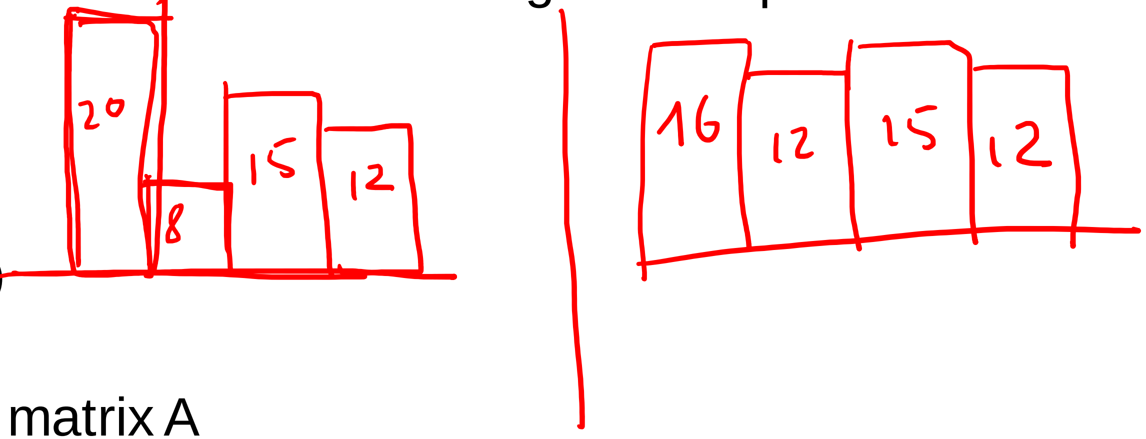



```
Try(k)
for v = 1 -> M-T-(n-k) do
    if check(v,k) then
        x_k = v;
        T = T + x_k;
        if k = n then Solution()
        else Try(k+1);
        T = T - x_k;

check(v,k)
    if k < n then return True;
    else return T + v = M;

main()
    T = 0;
    Try(1);
```

- The BACP is to design a balanced academic curriculum by assigning periods to courses in a way that the academic load of each period is balanced. There are N courses 1, 2, …, N that must be assigned to M periods 1, 2, …, M. Each course i has credit c_i and has some courses as prerequisites. The load of a period is defined to be the sum of credits of courses assigned to that period. The prerequisites information is represented by a matrix $A_{N\times N}$ in which $A_{i,j} = 1$ indicates that course i must be assigned to a period before the period to which the course j is assigned. Compute the solution satisfying constraints:
- Satisfy the prerequisites constraints: if $A_{i,j} = 1$, then course i must be assigned to a period before the period to which the course j is assigned
- The maximum load for all periods is minimal
- **Input**
  - Line 1 contains N and M ($2 \le N \le 16$, $2 \le M \le 5$)
  - Line 2 contains c1, c2, …, cN
  - Line i+2 (i = 1,…, N) contains the ith line of the matrix A
- **Output**
  - Unique line contains that maximum load for all periods of the solution found



N môn 1, 2, ..., n, C(i) số tín chỉ môn i

M học kỳ 1, 2, ..., m.

Biến quyết định: $\boxed{x_1, x_2, \ldots, x_n}$

trong đó x_i là học kỳ mà môn i được xếp vào

load[1..m]: load[j] tổng số tín chỉ các môn xếp vào H/kỳ j

loadMin: hàm mục tiêu → minimize

A[i,j]=True: môn i phải được học trước môn j

```
check(v,k)
  for i = 1 → k-1 do
    if A[i,k] = True then
      if xi ≥ v then return false;
    else if A[k,i] = True then
      if v ≥ xi then return false;
  return True.
```

```
Try(k) // thử giá trị cho xk
for v = 1 → m do
  if check(v,k) then
    xk = v; // xếp môn k vào H/kỳ v
    load[v] += C(k); //update
    if k = n then solution();
    else
      if load[v] < loadMin then
        Try(k+1);
    load[v] -= C(k); // recover
```

```
solution()
  maxLoad = MAX(load[1], ... load[m])
  if maxLoad < loadMin then
    loadMin = maxLoad;
```