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

THUẬT TOÁN ỨNG DỤNG

KỸ THUẬT QUAY LUI, NHÁNH VÀ CẬN

ONE LOVE. ONE FUTURE.

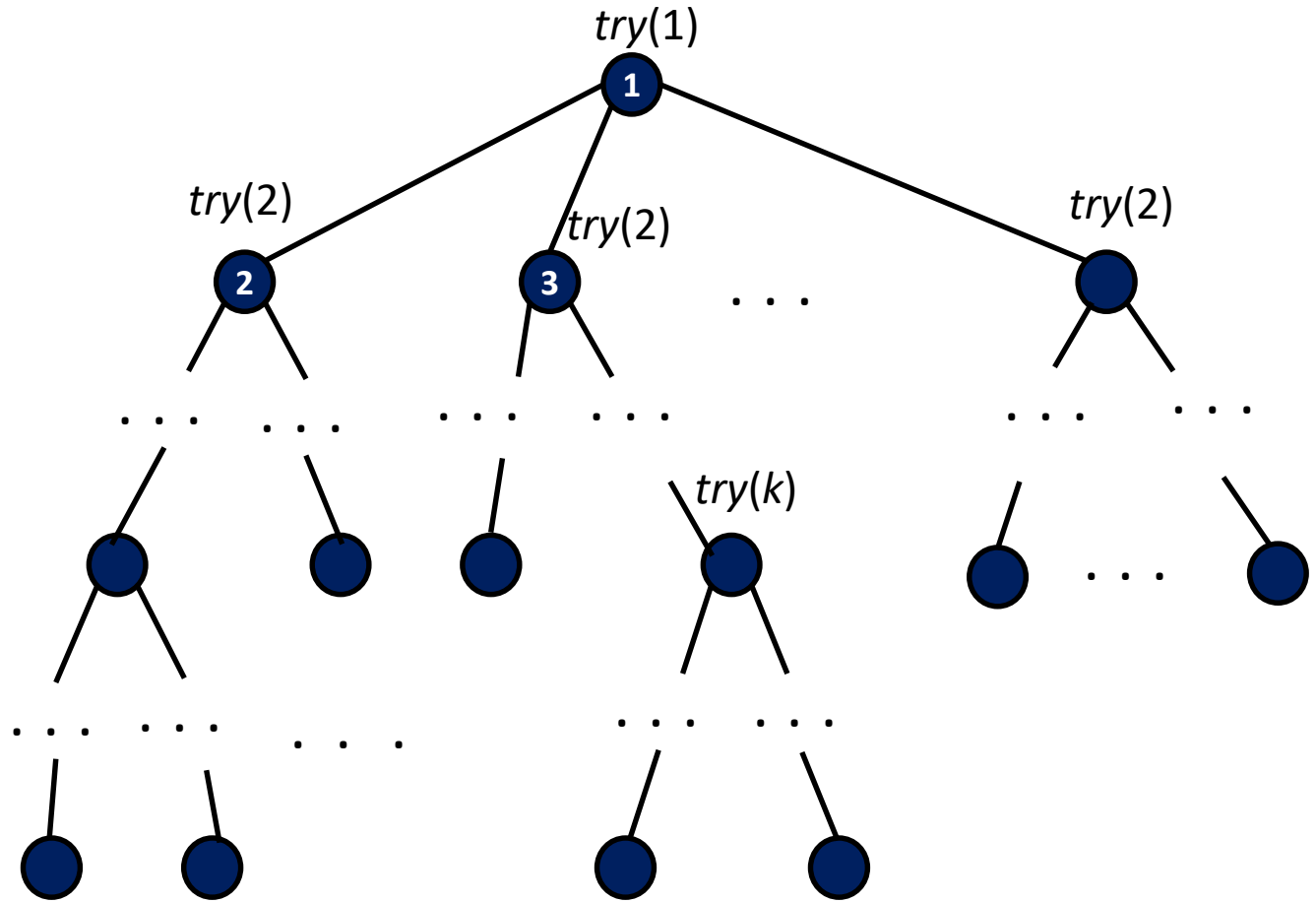
NỘI DUNG

- Sơ đồ chung nhánh và cận
- Bài toán lộ trình xe buýt đón trả khách
- Bài toán lộ trình xe tải giao hàng
- Bài toán cắt vật liệu 2D

SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN

- Thuật toán quay lui cho phép ta giải các bài toán liệt kê tổ hợp và bài toán tối ưu tổ hợp
- Phương án được mô hình hóa bằng một dãy các biến quyết định $X_1, X_2, \ldots, X_n$
- Cần tìm cho mỗi biến X_i một giá trị từ 1 tập rời rạc A_i cho trước sao cho
  - Các ràng buộc của bài toán được thỏa mãn
  - Tối ưu một hàm mục tiêu cho trước
- Tìm kiếm quay lui
  - Duyệt qua tất cả các biến (ví dụ thứ tự từ $X_1, X_2, \ldots, X_n$), với mỗi biến X_k
    - Duyệt lần lượt qua tất cả các giá trị có thể gán cho X_k, với mỗi giá trị v
      - Kiểm tra ràng buộc
      - Gán cho X_k
      - Nếu $k = n$ thì ghi nhận một phương án
      - Ngược lại, xét tiếp biến X_{k+1}

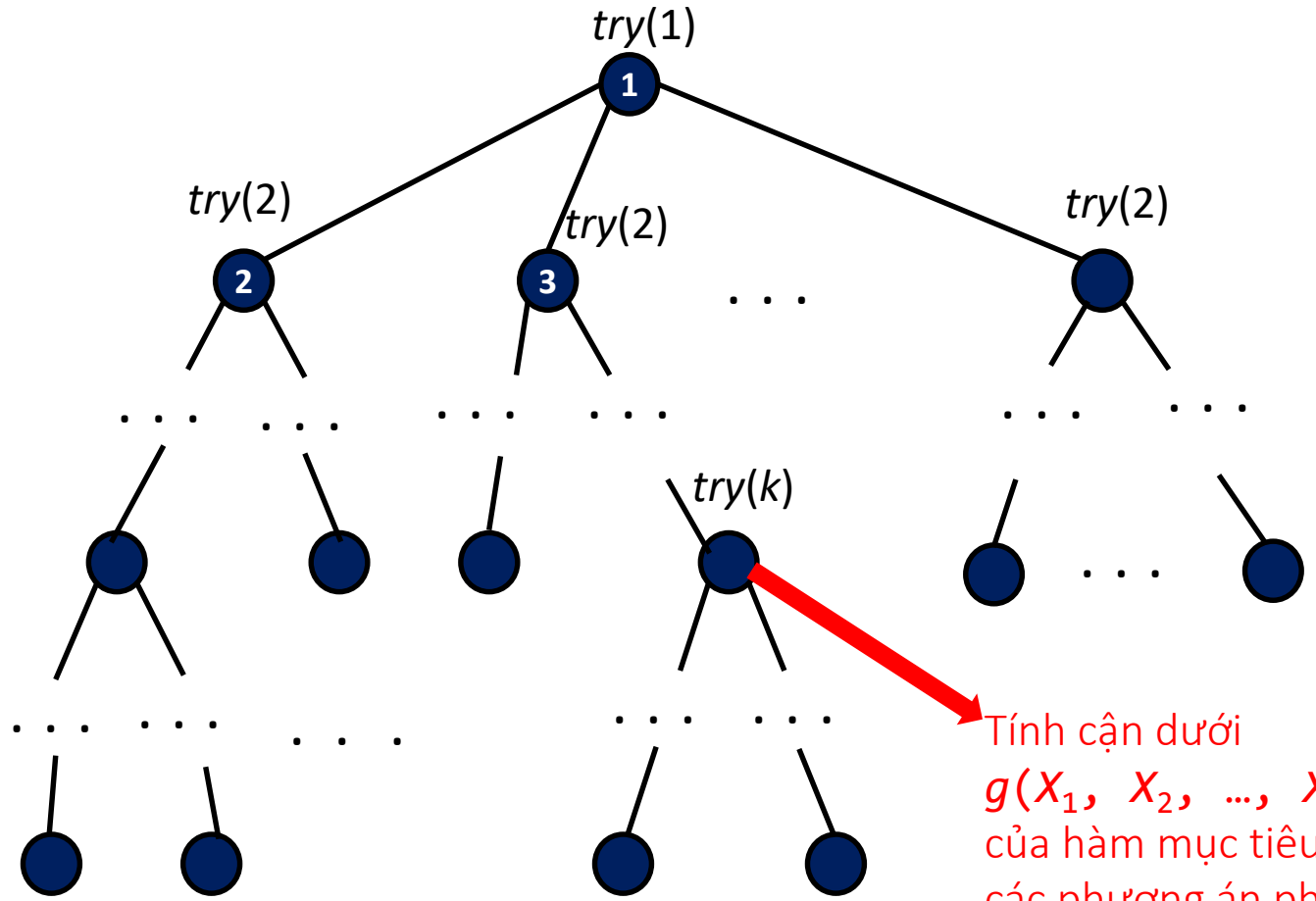
SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN



Bài toán liệt kê

```
try(k){ // thử các giá trị có thể gán cho X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then solution();
        else {
            try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

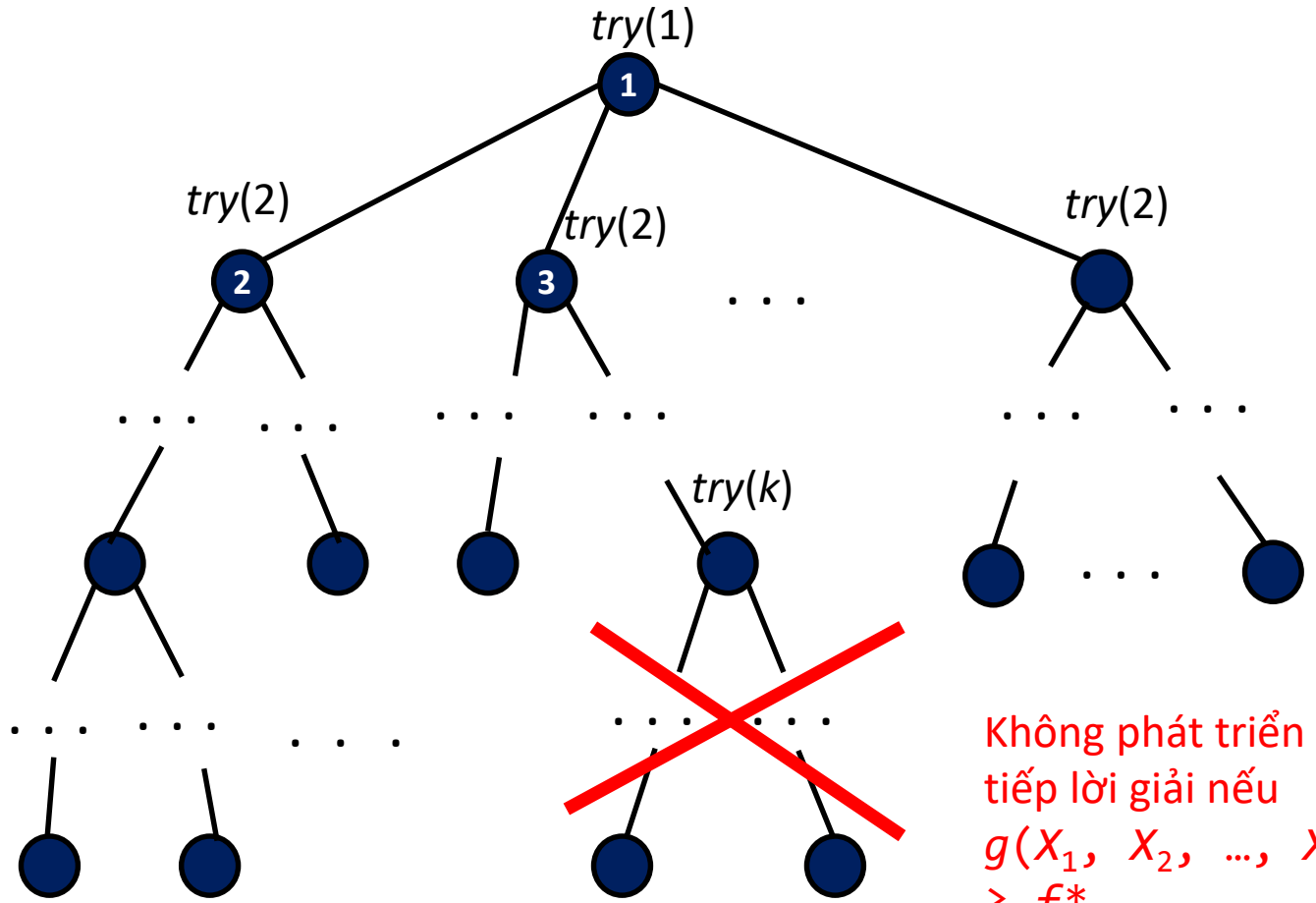
SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN



Bài toán tìm min (ký hiệu f^* : kỷ lục)

```
try(k){ // thử các giá trị có thể gán cho X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then updateBest();
        else {
            if g(X_1, X_2, …, X_k) < f* then
                try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

SƠ ĐỒ CHUNG QUAY LUI, NHÁNH VÀ CẬN



Không phát triển tiếp lời giải nếu $g(X_1, X_2, \ldots, X_k) \geq f^*$

Bài toán tìm min (ký hiệu f^* : kỷ lục)

```
try(k){ // thử các giá trị có thể gán cho X_k
  for v in A_k do {
     if check(v,k){
        X_k = v;
        [Update a data structure D]
        if k = n then updateBest();
        else {
            if g(X_1, X_2, …, X_k) < f* then
                try(k+1);
        }
        [Recover the data structure D]
     }
  }
}
```

BÀI TOÁN LỘ TRÌNH XE BUÝT

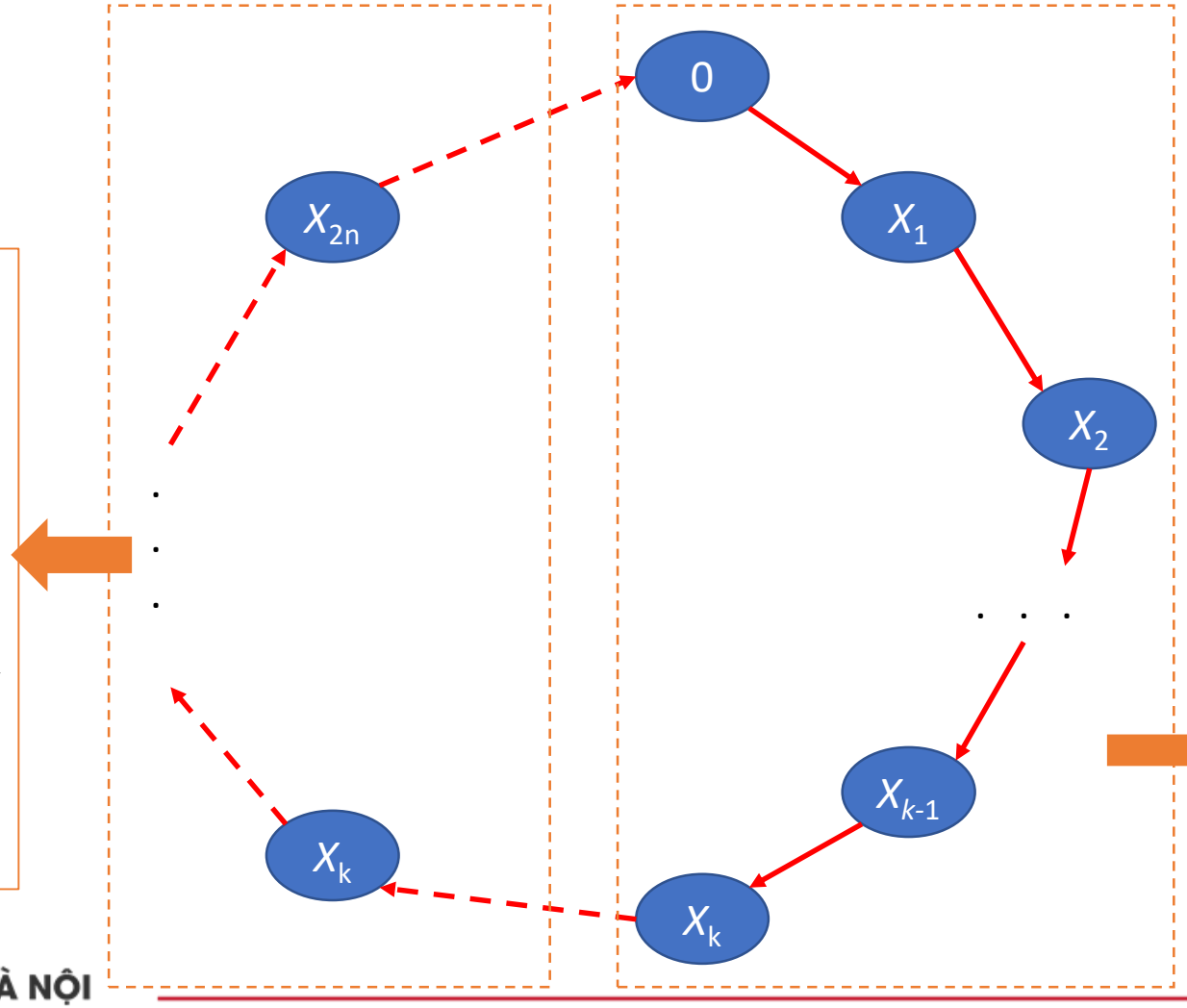
- Một xe buýt xuất phát từ điểm 0 cần được tính toán lộ trình để phục vụ đưa đón n khách và quay trở về điểm 0. Khách i có điểm đón là i và điểm trả là $i + n$ (i = 1, 2, ..., n). Xe buýt có K chỗ ngồi để phục vụ khách. Khoảng cách di chuyển từ điểm i đến điểm j là d(i, j), với i, j = 0, 1, 2, . . , 2n. Hãy tính lộ trình cho xe buýt sao cho tổng độ dài quãng đường di chuyển là nhỏ nhất, đồng thời số khách trên xe tại mỗi thời điểm không vượt quá K.

BÀI TOÁN LỘ TRÌNH XE BUÝT

- Một xe buýt xuất phát từ điểm 0 cần được tính toán lộ trình để phục vụ đưa đón n khách và quay trở về điểm 0. Khách i có điểm đón là i và điểm trả là $i + n$ (i = 1, 2, ..., n). Xe buýt có K chỗ ngồi để phục vụ khách. Khoảng cách di chuyển từ điểm i đến điểm j là d(i, j), với i, j = 0, 1, 2, . . , 2n. Hãy tính lộ trình cho xe buýt sao cho tổng độ dài quãng đường di chuyển là nhỏ nhất, đồng thời số khách trên xe tại mỗi thời điểm không vượt quá K.
- Thuật toán nhánh và cận
  - Mô hình hóa: $X_1, X_2, . . ., X_{2n}$ là dãy các điểm đón-trả trên lộ trình xe buýt (là hoán vị của 1, 2, ..., 2n).
  - *Cmin*: khoảng cách nhỏ nhất trong số các khoảng cách giữa 2 điểm
  - Mảng đánh dấu: visited[v] = true có nghĩa điểm v đã xuất hiện trên lộ trình và visited[v] = false, ngược lại
  - load: số khách đang có mặt trên xe
    - Khi lộ trình đi đến điểm đón thì load tăng lên 1 và khi đi đến điểm trả thì load giảm đi 1
  - f: độ dài của lộ trình bộ phận
  - f^*: độ dài lộ trình ngắn nhất đã tìm thấy (kỷ lục)

BÀI TOÁN LỘ TRÌNH XE BUÝT

- **Phân tích cận dưới**



- Lộ trình chưa đi qua, gồm $2n+1-k$ chặng, mỗi chặng có độ dài lớn hơn hoặc bằng *Cmin*
- Độ dài lộ trình đầy đủ phát triển tiếp từ X_k sẽ lớn hơn hoặc bằng $f + Cmin*(2n+1-k)$

Lộ trình bộ phận đã đi qua
- k chặng
- Độ dài f

BÀI TOÁN LỘ TRÌNH XE BUÝT

```
try(k){
  for v = 1 to 2n do {
     if check(v,k){
        X_k = v;
        f = f + d(X_{k-1},X_k); visited[v] = true;
        if v ≤ n then load += 1; else load -= 1;
        if k =2 n then updateBest();
        else {
            if f + Cmin*(2n+1-k) < f* then
                try(k+1);
        }
        if v ≤ n then load -= 1; else load += 1;
        f = f - d(X_{k-1},X_k); visited[v] = false;
     }
  }
}
```

```
check(v,k){
  if visited[v] = true then return false;
  if v > n then {
    if visited[v-n] = false then return false;
  }else{
    if load + 1 > K then return false;
  }
  return true;
}
```

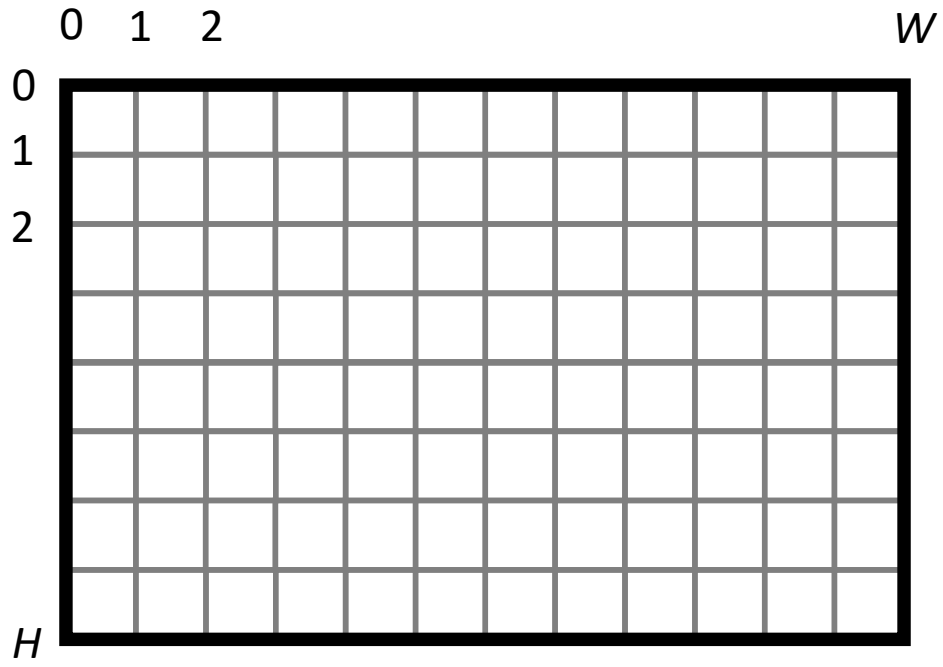
```
updateBest(){
  if f + d(X_{2n},0) < f* then {
    f* = f + d(X_{2n},0);
  }
}
```

BÀI TOÁN CẮT VẬT LIỆU 2D

- Một tấm phôi kim loại hình chữ nhật chiều dài là H và chiều rộng là W cần được cắt ra thành n miếng hình chữ nhật, miếng thứ i có chiều dài là $h[i]$ và chiều rộng là $w[i]$ trong đó $H, W, h[1], w[1], h[2], w[2], \ldots, h[n], w[n]$ là các số nguyên dương. Hãy tìm cách cắt thỏa mãn yêu cầu đặt ra.

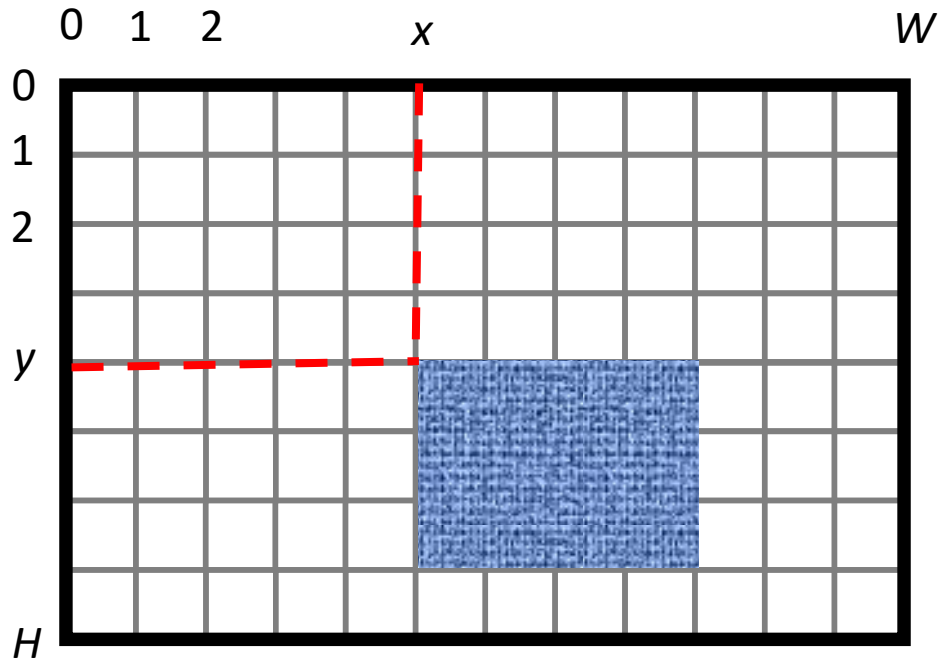
BÀI TOÁN CẮT VẬT LIỆU 2D

- Một tấm phôi kim loại hình chữ nhật chiều dài là *H* và chiều rộng là *W* cần được cắt ra thành *n* miếng thành phẩm hình chữ nhật, miếng thứ *i* có chiều dài là *h*[*i*] và chiều rộng là *w*[*i*] trong đó *H, W*, *h*[1], *w*[1], *h*[2], *w*[2], . . ., *h*[*n*], *w*[*n*] là các số nguyên dương. Hãy tìm cách cắt thỏa mãn yêu cầu đặt ra.
- Chia miếng phôi kim loại ban đầu thành lưới (các ô vuông đơn vị) bởi các đường thẳng song song với trục ngang và trục dọc cách đều nhau 1 đơn vị



BÀI TOÁN CẮT VẬT LIỆU 2D

- Một tấm phôi kim loại hình chữ nhật chiều dài là *H* và chiều rộng là *W* cần được cắt ra thành *n* miếng thành phẩm hình chữ nhật, miếng thứ *i* có chiều dài là *h*[*i*] và chiều rộng là *w*[*i*] trong đó *H, W*, *h*[1], *w*[1], *h*[2], *w*[2], . . ., *h*[*n*], *w*[*n*] là các số nguyên dương. Hãy tìm cách cắt thỏa mãn yêu cầu đặt ra.
- Chia miếng phôi kim loại ban đầu thành lưới bởi các đường thẳng song song với trục ngang và trục dọc cách đều nhau 1 đơn vị



Mỗi miếng hình chữ nhật con cần cắt ra sẽ nằm ở 1 vị trí nào đó trên lưới và được đặc trưng bởi tọa độ (x, y) của điểm góc trên bên trái và trạng thái xoay 90^0 hoặc không xoay

BÀI TOÁN CẮT VẬT LIỆU 2D

- Một tấm phôi kim loại hình chữ nhật chiều dài là H và chiều rộng là W cần được cắt ra thành n miếng thành phẩm hình chữ nhật, miếng thứ i có chiều dài là $h[i]$ và chiều rộng là $w[i]$ trong đó H, W, $h[1]$, $w[1]$, $h[2]$, $w[2]$, . . ., $h[n]$, $w[n]$ là các số nguyên dương. Hãy tìm cách cắt thỏa mãn yêu cầu đặt ra.
- Mô hình hóa: mỗi phương án cắt được biểu diễn bởi 1 dãy các bộ 3 biến quyết định ($x[1]$, $y[1]$, $o[1]$), ($x[2]$, $y[2]$, $o[2]$), . . . , ($x[n]$, $y[n]$, $o[n]$) trong đó (
  - ($x[i]$, $y[i]$): xác định vị trí của góc trên bên trái của hình chữ nhật con thứ i
  - $o[i]$: xác định trạng thái xoay 90^0 ($x[i] = 1$) hoặc không xoay ($x[i] = 0$)

BÀI TOÁN CẮT VẬT LIỆU 2D

- Một tấm phôi kim loại hình chữ nhật chiều dài là H và chiều rộng là W cần được cắt ra thành n miếng thành phẩm hình chữ nhật, miếng thứ i có chiều dài là $h[i]$ và chiều rộng là $w[i]$ trong đó H, W, $h[1]$, $w[1]$, $h[2]$, $w[2]$, . . ., $h[n]$, $w[n]$ là các số nguyên dương. Hãy tìm cách cắt thỏa mãn yêu cầu đặt ra.
- Mô hình hóa: mỗi phương án cắt được biểu diễn bởi 1 dãy các bộ 3 biến quyết định ($x[1]$, $y[1]$, $o[1]$), ($x[2]$, $y[2]$, $o[2]$), . . . , ($x[n]$, $y[n]$, $o[n]$) trong đó (
  - ($x[i]$, $y[i]$): xác định vị trí của góc trên bên trái của hình chữ nhật con thứ i
  - $o[i]$: xác định trạng thái xoay 90^0 ($x[i] = 1$) hoặc không xoay ($x[i] = 0$)
- Khi đã xác định đặt 1 miếng thành phẩm vào 1 vị trí nào đó trên miếng phôi thì cần đánh dấu vùng trên miếng phôi mà miếng thành phẩm đó đã chiếm chỗ để tránh 2 miếng thành phẩm đè lên nhau (overlap)

BÀI TOÁN CẮT VẬT LIỆU 2D

- Một tấm phôi kim loại hình chữ nhật chiều dài là H và chiều rộng là W cần được cắt ra thành n miếng thành phẩm hình chữ nhật, miếng thứ i có chiều dài là $h[i]$ và chiều rộng là $w[i]$ trong đó H, W, $h[1]$, $w[1]$, $h[2]$, $w[2]$, . . ., $h[n]$, $w[n]$ là các số nguyên dương. Hãy tìm cách cắt thỏa mãn yêu cầu đặt ra.
- Mô hình hóa: mỗi phương án cắt được biểu diễn bởi 1 dãy các bộ 3 biến quyết định ($x[1]$, $y[1]$, $o[1]$), ($x[2]$, $y[2]$, $o[2]$), . . . , ($x[n]$, $y[n]$, $o[n]$) trong đó (
  - ($x[i]$, $y[i]$): xác định vị trí của góc trên bên trái của hình chữ nhật con thứ i
  - $o[i]$: xác định trạng thái xoay 90^0 ($x[i] = 1$) hoặc không xoay ($x[i] = 0$)
- Khi đã xác định đặt 1 miếng thành phẩm vào 1 vị trí nào đó trên miếng phôi thì cần đánh dấu vùng trên miếng phôi mà miếng thành phẩm đó đã chiếm chỗ để tránh 2 miếng thành phẩm đè lên nhau (overlap)

→ Đánh dấu: mark[i, j] = true có nghĩa ô vuông đơn vị có tọa độ đỉnh góc trên bên trái (i, j) đã bị chiếm chỗ (i = 0, 1, …, H-1 và j = 0, 1, . . ., W-1)

BÀI TOÁN CẮT VẬT LIỆU 2D

- Hàm Try(k): thử giá trị cho x[k], y[k], o[k]

```
Try(k){
  for vo = 0 to 1 do {
    wk = w[k]; hk = h[k];
    if vo = 1 then { wk = h[k]; hk = w[k]; }
    for vx = 0 to W – wk do {
      for vy = 0 to H – hk do {
        if check(vo,vx,vy,k) then {
          x[k] = vx; y[k] = vy; o[k] = vo;
          doMark(vo,vx,vy,k, true); // do mark
          if (k = n) then solution(); else Try(k+1);
          doMark(vo,vx,vy,k, false); // undo mark
        }
      }
    }
  }
}
```

BÀI TOÁN CẮT VẬT LIỆU 2D

```
check(vo, vx, vy, k){
  wk = w[k]; hk = h[k];
  if(vo == 1){
    wk = h[k]; hk = w[k];
  }
  if(vx + h[k] > W) return false;
  if(vy + w[k] > H) return false;
  for(i = vx; i <= vx + wk - 1; i++)
    for(j = vy; j <= vy + hk - 1; j++)
      if(mark[i][j]) return false;
  return true;
}
```

```
doMark(vx, vy, vo, k, mark_value){
  wk = w[k]; hk = h[k];
  if(vo == 1){
    wk = h[k]; hk = w[k];
  }
  for(i = vx; i <= vx + h[k] - 1; i++)
     for(j = vy; j <= vy + w[k] - 1; j++)
         mark[i][j] = mark_value;
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !