CHIẾN LƯỢC QUAY LUI VÀ NHÁNH CẬN

- Chiến lược quay lui
- Chiến lược nhánh cận

CHIẾN LƯỢC QUAY LUI

- Các đặc trưng cơ bản
- Các ví dụ minh họa

CÁC ĐẶC TRƯNG CƠ BẢN

- Giải thuật được thiết kế bằng chiến lược quay lui biểu diễn nghiệm bài toán như một vector $x = (x_1, x_2, \ldots, x_n)$ với x_i được chọn trong tập A_i nào đó
- Quá trình tìm $x = (x_1, x_2, \ldots, x_n)$ được thực hiện bằng cách tính toán mỗi thành phần $x_1, x_2, \ldots, x_n$ tại một thời điểm

CÁC ĐẶC TRƯNG CƠ BẢN

- Tại bước thứ i, vector con $(x_1, x_2, ..., x_i)$, tìm được gọi là một nghiệm bộ phận của bài toán
- Để tìm thành phần thứ k của nghiệm (thỏa một số ràng buộc nào đó), khi đã chọn được k-1 thành phần $x_1, x_2, ..., x_{k-1}$ của x, giải thuật chọn (tính) x_k trong số các khả năng có thể có của nó trong A_k

CÁC ĐẶC TRƯNG CƠ BẢN

- Với mỗi khả năng j, giải thuật kiểm tra xem có chấp nhận được x_k không:
  - Nếu chấp nhận thì xác định x_k theo j, khi k = n thì có một lời giải, ngược thì tiếp tục xác định x_{k+1} (thường bằng đệ qui)
  - Nếu thử tất cả các khả năng $x_k \in A_k$ mà không có khả năng nào được chấp nhận thì quay lui lại bước trước để xác định lại x_{k-1}

CÁC ĐẶC TRƯNG CƠ BẢN

```
BackTracking(x[1..k], n) // xác định x[k], k nguyên
1 for j ←1 to n_k  // xét khả năng j, trong n_k khả năng
2     do  if accepting j
3             then  <computing x[k] in A_k that subjects to j>
4                     if k = n
5                         then < recording 1 solution >
6                         else BackTracking(x[1..k+1], n)
```

CÁC ĐẶC TRƯNG CƠ BẢN

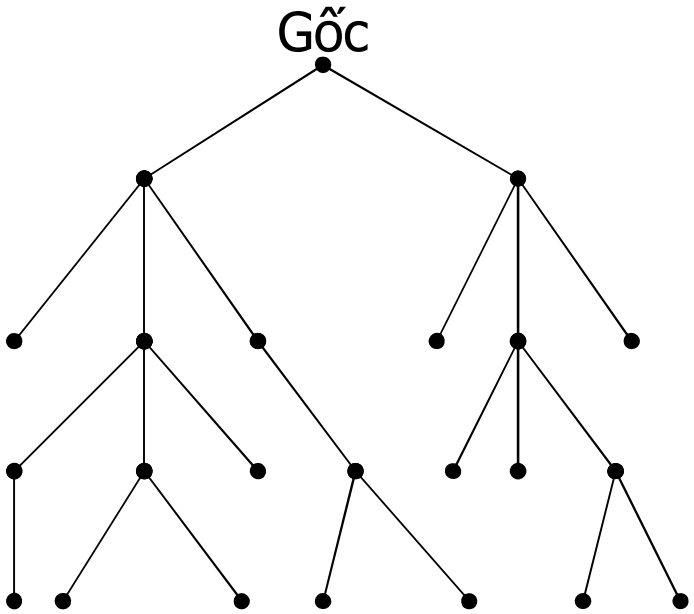
- Cần chỉ rõ tập khả năng {1, …, n_k} và cách kiểm tra ràng buộc <accepting j>
- Ngoài sự phụ thuộc vào j, các x_i còn phụ thuộc vào việc chọn các thành phần ở các bước trước

CÁC ĐẶC TRƯNG CƠ BẢN

- Giải thuật sinh ra một cây được gọi là cây không gian trạng thái (state-space tree)
- Trong nhiều bài toán, có thể phải ghi nhớ trạng thái của quá trình tìm kiếm khi xác định x_k theo j và trả lại trạng thái cũ sau lời gọi BackTracking(x[1..k+1], n)

CÁC ĐẶC TRƯNG CƠ BẢN

Cây không gian trạng thái của giải thuật quay lui



Khả năng chọn x_1

Khả năng chọn x_2 khi đã chọn x_1

Khả năng chọn x_3 khi đã chọn x_1, x_2

CÁC ĐẶC TRƯNG CƠ BẢN

- Khác với chiến lược vét cạn, chiến lược quay lui không tìm kiếm và phát triển mọi nghiệm bộ phận có thể dẫn đến nghiệm bài toán mà chỉ xét các nghiệm bộ phận thỏa mãn ràng buộc, có thể dẫn tới nghiệm bài toán
- Chiến lược quay lui tìm kiếm theo chiều sâu (depth-first seach)

CÁC VÍ DỤ MINH HỌA

- Tìm tất cả các xâu nhị phân n bits
- Tìm tất cả các chỉnh hợp chập k của n số {1, 2, …, n}
- Bài toán sắp đặt n quân hậu

TÌM TẤT CẢ CÁC XÂU NHỊ PHÂN n BITS

- Biểu diễn mỗi xâu nhị phân cần tìm là $b = (b_1, b_2, ..., b_n)$, với $b_i = 0,1$
- Giải thuật xác định các b_k, k=1, 2, …, n với các giá trị có thể chọn trong $A_k = \{0,1\}$
- Các lựa chọn cho $b_k \in \{0,1\}$ mặc nhiên được chấp nhận vì không có bất kỳ ràng buộc nào

TÌM TẤT CẢ CÁC XÂU NHỊ PHÂN n BITS

```
BanaryBackTracking(b[1..k], n)
1  for j ← 0 to 1  // đồng nhất các khả năng j với các giá trị
2     do    b[k] ←  j              // b_k có thể nhận trong A_k={0,1}
3           if k = n
4                then Print(b[1..k])
5                else BanaryBackTracking(b[1..k+1], n)
```

TÌM TẤT CẢ CÁC XÂU NHỊ PHÂN n BITS

- Giải thuật xác định $b=(b_1, b_2, ..., b_n)$ bắt đầu bằng việc xác định b_1 và gọi 2 lần đệ qui để tính b_2
- Kích thước đầu vào là n, nên

$$T(n)=2T(n-1)+O(1) = O(2^n)$$

TÌM TẤT CẢ CÁC CHỈNH HỢP CHẬP k CỦA n SỐ {1, 2, …, n}

- Biểu diễn mỗi chỉnh hợp chập k của n là $p = (p_1, p_2, \ldots, p_k)$ trong đó $p_i \in A_i = \{1, 2, \ldots, n\}$, và $p_i \neq p_j$
  - p_i nhận giá trị theo $j \in A_i$ với ràng buộc $p_i \neq p_1, p_2, \ldots, p_{i-1}$
  - Dùng biến logic b_j, ứng với khả năng chọn j =1,..., n, để ghi nhận j đã được gán cho một p_i trong chỉnh hợp
  - Các b_j được khởi tạo là true (j chưa sử dụng) và được gán bằng false nếu đã sử dụng j

TÌM TẤT CẢ CÁC CHỈNH HỢP CHẬP k CỦA n SỐ {1, 2, …, n}

```
K_PermutationBackTracking(p[1..i], k, n)
1  for j←1 to n  // j ∈ Ai={1, 2, …, n} that p[i] can accept
2    do if b[j] = true      // accepting possibility j
3          then p[i] ← j
4               b[j] ← false        // record new status
5               if i =k
6                  then Print_result(p[1..k])
7                  else K_PermutationBackTracking (p[1..i+1], k, n)
8               b[j] ← true         // return old status
```

TÌM TẤT CẢ CÁC CHỈNH HỢP CHẬP k CỦA n SỐ {1, 2, …, n}

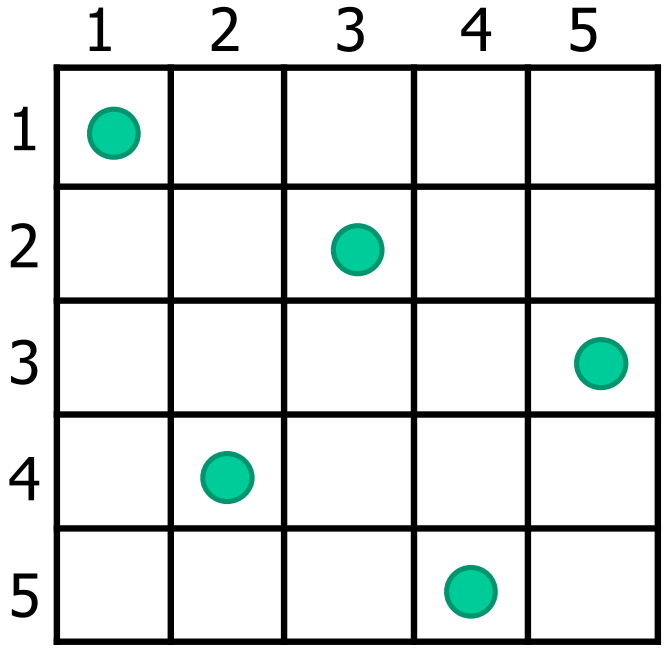
- Giải thuật xác định $p=(p_1, p_2, \ldots, p_k)$ bắt đầu bằng việc xác định p_1 và gọi n lần đệ qui (kích thước k-1) để tính p_2
- Kích thước đầu vào là k và n, nên

$$T(k, n)=nT(k-1, n)+O(n) = O(n^k)$$

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

Bài toán Cho bàn cờ vua n×n, hãy tìm cách đặt n quân hậu lên bàn cờ sao cho chúng không thể tấn công nhau

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU



Một (nghiệm) lời giải bài toán

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

- Một quân hậu chỉ được đặt tại ô(i, j) nếu chưa có quân hậu nào đặt ở hàng i hoặc cột j hoặc trên 2 đường chéo đi qua ô(i, j)
- Coi mỗi lời giải là một vector $x=(x_1, x_2, ..., x_n)$
- $x_i=j$ biểu diễn quân hậu ở hàng i được đặt trong cột j (đảm bảo mỗi quân hậu chỉ đặt trong một hàng)

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

- Một quân hậu chỉ được đặt tại ô(i, j), (nghĩa là $x_i = j$, với j= 1, 2, ..., n) nếu (trước đó) chưa có quân hậu nào được đặt ở cột j
- Dùng biến logic a[j] =true ghi nhận cột thứ j chưa có quân hậu nào

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

- Có 2n-1 đường chéo đi từ dưới lên trên, từ trái sang phải, mỗi ô(i, j) trên một đường chéo này có tổng i+j không đổi
  - Đường chéo (thứ nhất) mỗi ô có tổng i+j=2 (nhỏ nhất) là đường chéo chỉ đi qua ô(1, 1)
  - Đường chéo (thứ 2) mỗi ô có tổng i+j=3 là đường chéo đi qua ô(2, 1) và ô(1,2),
  - Đường chéo (thứ 2n-1) mỗi ô có tổng i+j=2n chỉ đi qua ô(n,n)
  - Dùng biến logic b[k]=true ghi nhận các ô(i, j) trên đường chéo có tổng i+j=k chưa có quân hậu nào

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

- Có 2n-1 đường chéo đi từ trên xuống, từ trái sang phải, mỗi ô(i, j) trên một đường chéo này có hiệu i-j không đổi
  - Đường chéo (thứ nhất) mỗi ô có hiệu i-j=1-n (nhỏ nhất) là đường chéo chỉ đi qua ô(1, n)
  - Đường chéo (thứ 2) mỗi ô có hiệu i-j=2-n là đường chéo đi qua ô(1,n-1) và ô(2, n),
  - Đường chéo (thứ 2n-1) mỗi ô có hiệu i-j=n-1 chỉ đi qua ô(n,1)
  - Dùng biến logic c[k]=true ghi nhận các ô(i, j) trên đường chéo có tổng i-j=k chưa có quân hậu nào

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

```
NQeen(x[1..i], n) // tìm x[i]
1  for j←1 to n  // đăt quân hậu hàng i vào cột j
2     do if a[j]=true and b[i+j]=true and c[i-j]=true
3           then    x[i] ← j
4                   a[j] ← false; b[i+j] ← false; c[i-j] ← fasle
5                    if i=n
6                         then Print_result(x[1..i])
7                         else NQeen(x[1..i+1], n)
8                   a[j] ← true; b[i+j] ← true; c[i-j] ← true
```

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

- Lưu ý: trước khi giải thuật thực thi cần gán các giá trị
  a[j]=true, ∀ j=1,.., n; b[k]=true, ∀ k=2,.., 2n;
  c[k]=true, ∀ k=1-n,.., n-1

BÀI TOÁN SẮP ĐẶT n QUÂN HẬU

- Giải thuật xác định $x=(x_1, x_2, ..., x_n)$ bắt đầu bằng việc xác định x_1 và gọi n lần đệ qui (kích thước n-1) để tính x_2
- Kích thước đầu vào là n, nên

$$T(n) = nT(n-1)+O(n) = O(n!)$$

BÀI TẬP VỀ NHÀ

- Đọc phần 12.1 Backtracking (trang 424) sách Levitin
- Bài tập thực hành: Hiện thực các giải thuật tìm các chỉnh hợp (hoán vị) và n quân hậu.

CHIẾN LƯỢC NHÁNH CẬN

- Các đặc trưng cơ bản
- Các ví dụ minh họa

CÁC ĐẶC TRƯNG CƠ BẢN

- Chiến lược quay lui cho phép tìm được nghiệm đúng của bài toán, nhưng do phải thử mọi khả năng, khi kích thước bài toán lớn sẽ rất kém hiệu quả
- Chiến lược nhánh cận khắc phục được hạn chế này bằng cách, xác định nhánh cận với mục tiêu tại mỗi bước tìm kiếm, vì vậy loại bỏ được hầu hết các hướng tìm kiếm không cần thiết (trong cây không gian tìm kiếm)

CÁC ĐẶC TRƯNG CƠ BẢN

- Chiến lược nhánh cận thường được áp dụng để giải các bài toán tối ưu
- Chiến lược nhánh cận dựa trên chiến lược quay lui và một hàm lượng giá mục tiêu hướng đến các nhánh cận mục tiêu để tìm lời giải nhanh nhất có thể

CÁC ĐẶC TRƯNG CƠ BẢN

- Chiến lược nhánh cận được áp dụng để giải các bài toán tối ưu với mô hình

  Tìm $\min\{f(x) \mid x=(x_1, x_2, \ldots, x_n) \in D=A_1 \times A_2 \times \ldots \times A_n\}$

- f(x) gọi là hàm mục tiêu

CÁC ĐẶC TRƯNG CƠ BẢN

- Một nghiệm của bài toán là một bộ $(x_1, x_2, ..., x_n) \in D$ sao cho $f(x_1, x_2, ..., x_n)$ nhỏ nhất
- Một bộ k thành phần $(a_1, a_2, ..., a_k)$, $a_i \in A_i$ gọi là một nghiệm (phương án) bộ phận cấp k

CÁC ĐẶC TRƯNG CƠ BẢN

- Nếu có một hàm g xác định trên tập tất cả các nghiệm bộ phận của bài toán và thỏa:

  $g(a_1, a_2, ..., a_k) \leq \min\{f(x) \mid x \in D, x_i = a_i, i = 1, 2, ..., k\}$,

  với mọi $(a_1, a_2, ..., a_k)$, k=1,2, … (1)

  thì $g(a_1, a_2, ..., a_k)$ được gọi là cận dưới hàm mục tiêu trên tập $D = \{(a_1, a_2, ..., a_k) \mid a_i \in A_i, k = 1, 2, ...\}$

CÁC ĐẶC TRƯNG CƠ BẢN

- Giả sử x* là một phương án với giá trị hàm mục tiêu nhỏ nhất trong các phương án đã tìm được, ký hiệu f*=f(x*)
- Ta nói x* là phương án tốt nhất hiện có, còn f* là kỷ lục (record)

CÁC ĐẶC TRƯNG CƠ BẢN

- Nếu $g(a_1, a_2, ..., a_k) > f^*$ thì từ (1) ta có

  $f^* < g(a_1, a_2, ..., a_k) \leq \min\{f(x) \mid x \in D, x_i = a_i, i = 1, 2, ..., k\}$
- Suy ra tập con các phương án trong $D(a_1, a_2, ..., a_k)$ chắc chắn không chứa phương án tối ưu
- Do đó loại các phương án trong $D(a_1, a_2, ..., a_k)$ khỏi quá trình tìm kiếm

CÁC ĐẶC TRƯNG CƠ BẢN

BranchBound(x[1..k], n) // Developing (a_1, a_2, …., a_{k-1})

1 **for** $a_k \in A_k$

2 **do** **if** accepting a_k

3 **then** $x_k \leftarrow a_k$

4 **if** k = n

5 **then** < update record>

6 **else if** $g(a_1, a_2, ..., a_k) \leq f^*$

7 **then** BranchBound(x[1..k+1], n)

CÁC ĐẶC TRƯNG CƠ BẢN

- Có thể khởi tạo f* = +∞
- Sau khi thực hiện xong BranchBound(x[1]) và có được f*, nếu
  - f* < +∞ thì f* là giá trị tối ưu và x* là phương án tối ưu
  - Ngược lại bài toán không có nghiệm
- Để giải bài toán bằng kỹ thuật nhánh cận, điều quan trọng là phải tìm được hàm cận dưới g và hàm này càng lượng giá được giá trị nhỏ nhất của hàm mục tiêu càng tốt

CÁC VÍ DỤ MINH HỌA

- Bài toán người du lịch
- Bài toán cái túi (tham khảo)

BÀI TOÁN NGƯỜI DU LỊCH

- Một người du lịch muốn đi tham quan n thành phố $T_1,\ldots,T_n$. Xuất phát từ một thành phố nào đó, người du lịch muốn đi qua tất cả các thành phố còn lại, mỗi thành phố đúng một lần, rồi quay về thành phố xuất phát. Biết c_{ij} là chi phí từ thành phố T_i đến thành phố T_j. Tìm hành trình (một cách đi) có tổng chi phí nhỏ nhất

BÀI TOÁN NGƯỜI DU LỊCH

- Cố định thành phố xuất phát T_1, bài toán dẫn đến tìm cực tiểu của hàm

$$f(x_2, x_3, \ldots, x_n)=c[1, x_2]+c[x_2, x_3]+\ldots+c[x_n,1]$$

với điều kiện $(x_2, x_3, \ldots, x_n)$ là một hoán vị của 2, 3, …, n

BÀI TOÁN NGƯỜI DU LỊCH

- Ký hiệu $c_{min}=min\{c[i, j], i, j=1,2,...n, i\neq j\}$
- Giả sử đang có phương án bộ phận $(u_1, u_2, ..., u_k)$, thì hành trình bộ phận qua k thành phố là

$$T_1 \rightarrow T(u_2) \rightarrow ... \rightarrow T(u_{k-1}) \rightarrow T(u_k)$$

- Chi phí phải trả theo hành trình bộ phận là

$$\sigma=c[1, u_2]+c[u_2, u_3]+...+c[u_{k-1},u_k]$$

BÀI TOÁN NGƯỜI DU LỊCH

- Hành trình đầy đủ của hành trình bộ phận còn phải đi qua n-k thành phố còn lại rồi quay về T_1 bao gồm n-k+1 đoạn đường
- Do chi phí phải trả cho mỗi đoạn trong n-k+1 đoạn còn lại không ít hơn c_{min} nên cận dưới (hàm mục tiêu) trên tập các phương án bộ phận có thể tính theo công thức

$$g(u_1, u_2,\ldots u_k) = \sigma+(n-k+1)c_{min}$$

$$\leq \min\{c[1, u_2]+\ldots+c[u_{k-1},u_k]+c[x_k, x_{k+1}]+\ldots+c[x_n,1]\}$$

$$= \min\{f(x) \mid x\in D, x_i=u_i, i=1, 2,\ldots, k\}$$

BÀI TOÁN NGƯỜI DU LỊCH

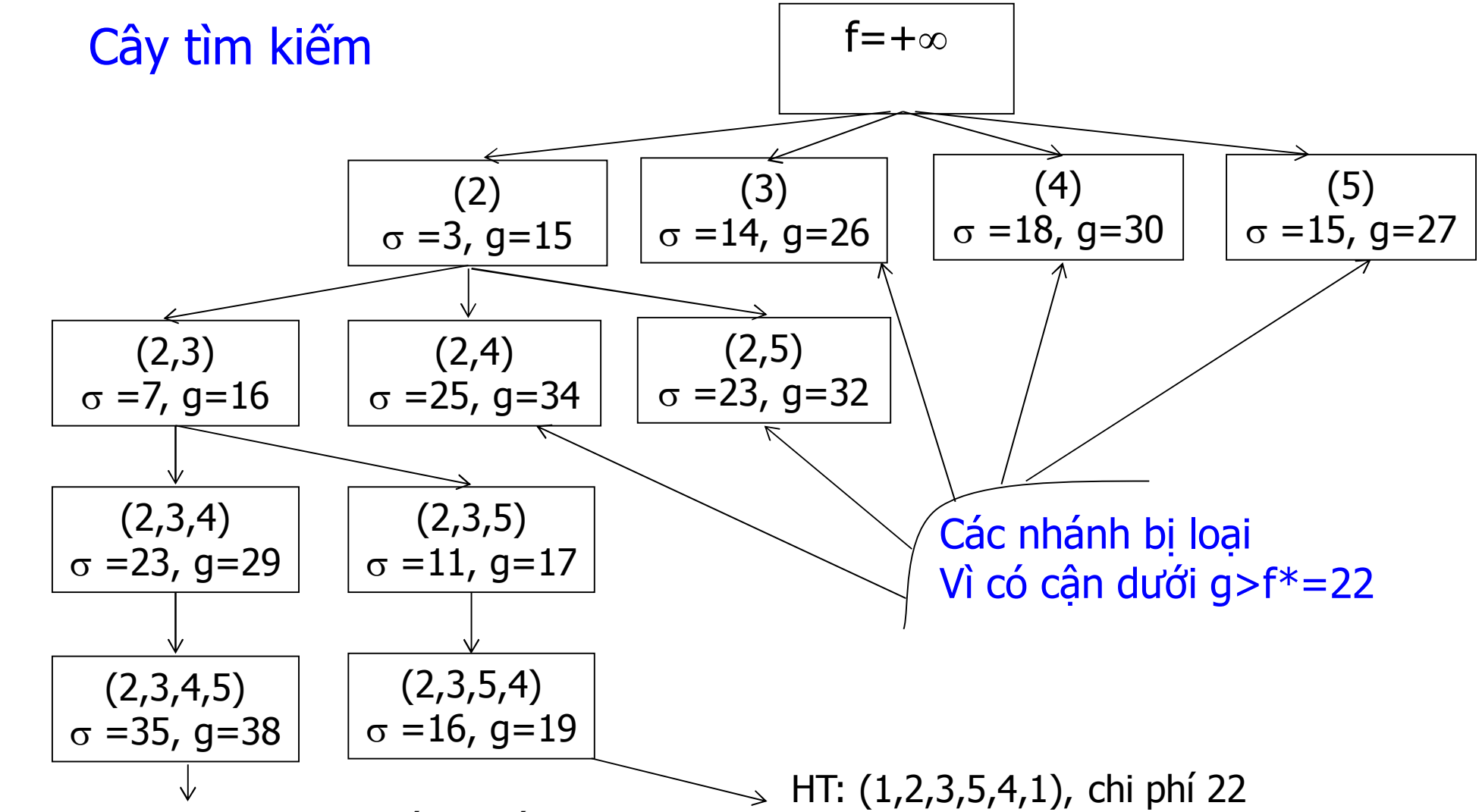
Giả sử ma trận chi phí đường đi như sau

$$C = \begin{bmatrix} 0 & 3 & 14 & 18 & 15 \\ 3 & 0 & 4 & 22 & 20 \\ 17 & 9 & 0 & 16 & 4 \\ 6 & 3 & 7 & 0 & 12 \\ 9 & 15 & 11 & 5 & 0 \end{bmatrix}$$

thì $c_{min}=3$

BÀI TOÁN NGƯỜI DU LỊCH

Cây tìm kiếm



BÀI TOÁN NGƯỜI DU LỊCH

- Kết thúc giải thuật, ta thu được phương án tối ưu (1,2,3,5,4,1) tương ứng với hành trình

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_5 \rightarrow T_4 \rightarrow T_1$$

và chi phí nhỏ nhất là 22

BÀI TẬP VỀ NHÀ

- Đọc phần 12.2 Branch-and-Bound (sách Levitin)
- Đọc bài toán cái túi (sách toán Toán rời rạc của Nguyễn Đức Nghĩa-Nguyễn Tô Thành)
- Hiện thực giải thuật giải bài toán người du lịch