**THAM LAM**

1. **Bài toán cái túi**
* Ý tưởng tham lam 1:

 Đồ vật có giá trị lớn còn lại được lấy trước.

Chi tiết:

* Sắp xếp các đồ vật theo thứ tự không tăng của giá trị.
* Chọn đồ vật từ đầu đến cuối (từ gtri cao đến gtri thap) nếu dung lượng còn lại của túi đủ chứa nó.
* Ý tưởng tham lam 2:

Đồ vật có trọng lượng nhỏ còn lại được lấy trước

Chi tiết:

* Sắp xếp các đồ vật theo thứ tự không giảm của trọng lượng.
* Chọn đồ vật từ đầu đến cuối (trọng lượng cao đến thấp) nếu dung lượng còn lại của túi đủ chứa nó.
* Ý tưởng tham lam 3:

Đồ vật có đơn giá lớn còn lại được lấy trước.

Chi tiết:

* Sắp xếp các đồ vật theo thứ tự không tăng của giá trị một đơn vị trọng lượng (gtri/trọng lượng).
* Chọn đồ vật từ đầu đến cuối.
1. **Bài toán người đi du lịch**
* Ý tưởng tham lam

Chọn thành phố gần nhất tính từ thành phố hiện thời.

Tổ chức dữ liệu: Đồ thị G= (V,E),V : tập đỉnh, E: tập cạnh. Mô tả đồ thị dạng ma trận kề.

1. **Đường đi ngắn nhất**
* **Bài toán**

Đồ thị G = (V,E), đơn đồ thị liên thông (vô hướng hoặc có hướng)

Có trọng số, V: tập đỉnh, E: tập cạnh

Tìm đường đi ngắn nhất từ s0∈V đến tất cả các đỉnh còn lại.

* **Thuật toán Dijkstra**

Có đồ thị G=(V,E), s0.

* L(v): độ dài đường đi ngắn nhất từ s0 đến đỉnh v (gọi là nhãn của v)
* Gọi S là tập đỉnh đã xét.
* Khởi tạo: S={s0}, L{s0}=0, L(v)=∞ ∀v∈V\S.
* Tại mỗi bước lặp:

Cập nhật lại nhãn các đỉnh thuộc V\S (tập V trừ tập S)

Tìm đỉnh thuộc tập V\S có nhãn nhỏ nhất (tham lam) kề với S để đưa vào S.

1. **Cây bao trùm nhỏ nhất**
* **Bài toán**

Cho đơn đồ thị G=(V,E) , V: tập các đỉnh, E: tập các cạnh

Cây T gọi là cây bao trùm của G nếu T là đồ thị con của G và chứa tất cả các đỉnh thuộc G (có số đỉnh =V).

* **Thuật toán Prim**

T = G(VT,ET) là cây khung tối thiểu cần tìm.

Ý tưởng:

* Chọn 1 đỉnh tùy ý vào VT.
* Khi |VT| < |V|

+ Tìm cạnh (s,t) ,s∈VT, t∈V\VT có trọng số nhỏ nhất (nối VT và \VT).

 + Thêm đỉnh t vào VT, (s,t) vào ET.

* **Thuật toán Kruskal**

T = G(VT,ET) là cây khung tối thiểu cần tìm.

Khi G có n đỉnh thì T có n-1 cạnh.

Ý tưởng: Xây dựng tập n-1 cạnh của T theo nguyên tắc.

* Khởi tạo ET={}, VT=V
* Xét lần lượt các cạnh có trọng số nhỏ đến lớn nếu không tạo thành chu trình trong trong T thì thêm cạnh đó vào VT.

**QUY HOẶC ĐỘNG**

1. **Bài toán Fibonaci**

Tính Fibonaci bằng QHĐ

Phân rã:

F(n) = F(n-1) + F(n-2)

Giải bài toán con

 F(0) = 0

 F(1) = 1

Tổng hợp

 F(n) = F(n-1) + F(n-2)

1. **Bài toán cái túi**
* Có n đồ vật, b trọng lượng túi
* Phân rã:

Với giá trị i (1..n) và L (0…b) Gọi MaxV(i,L) là tổng giá trị lớn nhất có thể chọn trong i đồ vật với trọng lượng tối đa cảu túi là L. Khi đó MaxV(n,b) là giá trị lớn nhất mang đi được.

* Giải bài toán con: MaxV(0,L) = 0 với mọi L, và MaxV(I,0) = 0 với mọi i.
* Tổng hợp:
* Đã có MaxV(i-1,L): giá trị lớn nhất mang đi được với i-1 đồ vật khi trọng lượng túi là L.
* Xét đồ vật thứ i trọng lượng túi vẫn là L:

Chỉ mang thêm đồ vật thứ i khi giá trị của túi lúc mang i-1 đồ vật ở trọng lượng túi là L-w[i] cộng với giá trị của đồ vật thứ i, c[i], lớn hơn khi không mang đồ vật thứ i, MaxV(i-1,L).

**MaxV(i,L) = Max{MaxV(i-1,L-w[i]) + c[i], MaxV(i-1,L)}**

1. **Bài toán tìm dãy con có tổng lớn nhất**
* Ý tưởng qui hoặc động

Phân rã:

* Gọi MaxS[i] là tổng lớn nhất của dãy con liên tiếp có i phần tử a[1]….a[i]
* Khi đó MaxS[n] là giá trị lớn nhất của dãy con liên tiếp cần tìm.

Tổng hợp:

* Giả sử i>1 và MaxS[k] là đã biết với k=1…, i-1
* Ta cần tính MaxS[i] là tổng của dãy con liên tiếp lớn nhất của dãy này có thể là một trong 2 trường hợp.

TH1: Các dãy con liên tiếp có chứa a[i]

TH2: Các dãy con liên tiếp không chứa a[i]

* Gọi MaxE[i] là tổng lớn nhất của các dãy con liên tiếp của dãy a[1]…..a[i] chứa chính a[i].
* Tổng lớn nhất của các dãy con liên tiếp của dãy a[1]…a[i-1], nghĩa là MaxS[i-1].
* **MaxS[i] = max{MaxS[i-1], MaxE[i]}**

Tính **MaxE[i]**

Để tính MaxE[i], i= 1,….,n ta có thể sử dụng công thức đệ quy như sau:

 Với i=1 thì MaxE[i]=a[1]

 Với i>1 Gọi C là dãy con kế liên tiếp lớn nhất dãy a[1]…a[i] có chứa a[i]. Có 2 khả năng:

 Nếu C chứa a[i-1] thì tổng lớn nhất là MaxE[i-1] + a[i].

 Nếu C không chứa a[i-1] thì C chỉ gồm a[i] và độ dài lớn nhất là a[i]

 **MaxE[i] = max{a[i], MaxE[i-1] + a[i]} i>1**

1. **Bài toán tìm xâu con chung dài nhất**

Ý tưởng

* Phân rã
* m chiều dài xâu X, n chiều dài xâu Y
* Với mỗi 0≤i≤m và 0≤j≤n gọi C[i,j] là độ dài cảu dãy con chung dài nhất cảu hai dãy

Xi= x1….xi và Yj= y1…yi

X0,y0 rỗng.

* Khi đó C[m,n] là chiều dài xâu con chung dài nhất của X và Y.
* Bài toán con: C[0,j]=0 ; j=1…n

 C[i,0]=0 ; i=1….m

* Với i>0, j>0 tính C[i,j]
* Nếu xi=yj thì dãy con chung dài nhất của xi, yj sẽ thu được bằng việc bổ sung xi(cũng là yi) vào dãy con chung dài nhất của 2 dãy Xi-1 và Yj-1.

**C[i,j] = C[i-1, j-1]+1**

* Nếu Xi # Yj thì dãy con chung dài nhất của Xi và Yj sẽ là dãy con dài hơn trong hai dãy con chung dài nhất của (Xi-1 và Yi) và của (Xi và Yj-1)

**C[i,j] = Max{C[i-1,j], C[i,j-1]}**

1. **Đường đi ngắn nhất**

Thuật toán Floyd

* Ý tưởng:
* Nếu k nằm trên đường đi ngắn nhất từ i đến j thì đường đi từ i đến k và từ k đến j cũng ngắn nhất.
* Phân rã:
* n số đỉnh của G
* Gọi d[i,j] là đường đi ngắn nhất từ đỉnh i đến đỉnh j
* Qui ước pk[i,j] với k=0…n lưu giá trị từ 0…k thể hiện đường đi ngắn nhất từ i đến j có qua đỉnh pk[i,j].
* n là số đỉnh G, d[i,j], pk[i,j]
* pk[i,j]=0 đường đi ngắn nhất từ i đến j không đi qua pk[i,j].
* pk[i,j]!=0 đường đi ngắn nhất từ i đến j đi qua pk[i,j]
* Khi k=n thì pk[i,j] cho biết đường đi cần tìm.
* Bài toán con
* d[i,j]=a[i,j]
* p0[i,j]=0
* Tổng hợp
* Nếu d[i,j] là đường đi ngắn nhất từ i đến j đã xét ở bước k (đã xét đi qua 1-k đỉnh)
* ở bước k+1:

d[i,j]= min(d[i,j], d[i,k+1]+ d[k+1,j])

1. **Cây nhị phân tìm kiếm**
* Phân rã
* Gọi Op(1…n) là số phép so sánh của cây nhị phân tìm kiếm tối ưu của mảng A[1..n]. Nếu A[r] là khóa của nút gốc, ta có:

Op(1..n) = Op(1…r-1) + Op(r+1…n) + SumF(1…n)

(SumF(1…n) = f[1] + f[2] +….+f[n])

Vì Op(1…n) là tối ưu nên ta có

Op(1…n) = min {Op(1…r-1) + Op(r+1…n): r=1..n} + SumF(1…n)

* Gọi C[i,j] là số phép so sánh của cây nhị phân tìm kiếm tối ưu cho mảng con A[i…j]
* Đặt F[i,j] = f[i] + f[i+1] +….+f[j]
* Ta có

C[i,j] = min{C[i,r-1] + C[r+1,j] : r=i…j} + F[i,j]

* Bài toán con

C[i,i] = F[i,i]

* Tổng hợp:

C[i,j] = min{C[i,r-1] + C[r+1,j]} + F[i,j]

Phần tính F[i,j] coi thêm nha!!!!!!!!!!!!!!!!!