Thuật toán ứng dụng
Bài thực hành số 6

Giảng viên: TS. Đinh Viết Sang
Trợ giảng: Nguyễn Trung Hiếu

Viện Công nghệ thông tin & truyền thông
Đại học Bách khoa Hà Nội

05/2021

Nội dung

07. CHANGE

- Cho các đồng tiền có mệnh giá lần lượt là \$1, \$5, \$10, \$50, \$100, \$500.
- Cần tìm cách sử dụng ít đồng tiền nhất để tạo ra tổng tiền N ($1 \leq N \leq 999$).

Thuật toán

- **Thuật toán 1:** Duyệt vét cạn tất cả các cách chia tiền, tìm cách có số lượng đồng tiền nhỏ nhất.
- **Thuật toán 2:** Tham lam: Xét lần lượt các mệnh giá từ lớn đến nhỏ, lấy tối đa số đồng tiền có thể để tổng tiền không vượt quá N. Cứ làm như vậy cho đến khi lấy đủ số tiền.

Tính đúng đắn

- Luôn tạo ra được tổng N: do khi xét mỗi mệnh giá, ta lấy tối đa có thể để tổng không vượt quá N, vậy ta luôn có tổng tiền $S \leq N$. Mà ta lại có mệnh giá \$1, nên sẽ tồn tại cách chọn để $S = N$.
- Cách chọn này là tối ưu: để ý rằng số đồng tiền \$1 được chọn < 5, do ngược lại ta có thể đổi 5 đồng \$1 lấy 1 đồng \$5. Tương tự số đồng \$5 < 2, ... $(*)$
- Giả sử cách chọn của chúng ta lấy a đồng \$500, 1 cách chọn tối ưu lấy $b < a$, $(b + a0 = a)$ đồng \$500. Ta có

$$N = a * 500 + a' = b * 500 + b' = (a - a0) * 500 + b'$$

với a', b' là số tiền tạo ra từ các tờ tiền nhỏ hơn. Nên: $a' + 500 \leq b'$, mà từ các đồng bé hơn \$500 không thể tạo ra tổng ≤ 500 được do $(*)$ nên không tồn tại b'. Vậy lấy a đồng là tối ưu.

Code

```
1 int a[6] = {1, 5, 10, 50, 100, 500};
2 int res = 0;
3 for (int i = 5; i >= 0; i--) {
4     res += n / a[i];
5     n %= a[i];
6 }
7 cout << res << endl;
```

07. PTREES

- Có N cái cây, cây thứ i cần t_i ngày để mọc
- Mỗi ngày trồng được một cây
- Hỏi ngày sớm nhất mà tất cả các cây đều mọc xong?

Thuật toán

- Cây càng mọc chậm thì càng phải trồng sớm
- Vì vậy ta sắp xếp các cây theo thứ tự mọc từ chậm đến nhanh, và trồng các cây theo thứ tự đó
- Cây thứ i sau khi sắp xếp sẽ được trồng ở ngày thứ i.

Code

```
8  int n;
9  vector<int> a;
10 cin >> n;
11 for (int i = 0; i < n; ++i) cin >> a[i];
12 sort(a.begin(), a.end(), greater<int>());
13 for (int i = 0; i < n; ++i) a[i] += i + 2;
14 cout << *max_element(a.begin(), a.end()) << endl;
```

Ôn tập

- Thuật toán trực tiếp
- Duyệt vét cạn - Đệ quy
- Chia để trị
- Quy hoạch động
- Đồ thị
- Tham lam

Duyệt vét cạn - Đệ quy

- Xác định hoán vị cần duyệt
  - Các biến để duyệt
  - Miền dữ liệu cho biến
- Điều kiện dừng khi đệ quy
- Sử dụng kỹ thuật nhánh-cận để giảm thời gian tìm kiếm
  - Tạo hàm đánh giá, so sánh với kết quả tốt nhất đã tìm được
  - Duyệt các hoán vị theo một chiến thuật thay vì ngẫu nhiên

Chia để trị

- Chia nhỏ bài toán thành các bài toán con không gối nhau
- Tìm kiếm nhị phân
  - Xác định không gian tìm kiếm (đoạn mà giá trị cần tìm có thể nằm trong)
  - Loại bỏ không gian thừa theo đúng hàm kiểm tra

Quy hoạch động

- Chia bài toán thành nhiều bài toán con gối nhau
- Xác định các biến quy hoạch động dựa vào các giới hạn đề bài, kinh nghiệm bản thân
- Tìm các trường hợp đặc biệt, khởi tạo
- Viết công thức truy hồi

Đồ thị

- Thuật toán tìm kiếm - DFS, BFS
- Thuật toán tìm đường đi ngắn nhất - Floyd, Ford-Bellman, Dijkstra
- Thuật toán tìm cây khung nhỏ nhất - Kruskal, Prim
- Thuật toán tìm thành phần liên thông mạnh - Tarjan

Tham lam

- Cần tìm chiến thuật tham lam hợp lý
- Hãy chứng minh nếu có thể để đảm bảo

Lưu ý khi lập trình

- Sử dụng thư viện **#include<bits/stdc++.h>**
- Luôn sử dụng câu lệnh **ios_base::sync_with_stdio(0)** với các bài có dữ liệu 10^5 số trở lên. (Đặt trước khi cin)
- Chú ý kiểu dữ liệu **int, long long**
- Khai báo mảng cần xác định rõ giới hạn của mảng và luôn khai báo thừa ra 1 vài ô nhớ. Ví dụ: mảng *a* cần chứa 100 phần tử thì nên khai báo **a[105]**.
- Hạn chế khai báo mảng trong chương trình con với các bài đệ quy, nên khai báo toàn cục.
- Nếu không nghĩ ra cách giải tối ưu thì hãy làm cách tốt nhất mình đã nghĩ ra dù không được số điểm tối đa.

Các cấu trúc dữ liệu-hàm trong C++

Cấu trúc dữ liệu:

- pair - lưu trữ 1 cặp dữ liệu
- vector - danh sách liên kết nhưng có tốc độ đọc O(1)
- list - danh sách liên kết
- priority_queue - cấu trúc heap

Hàm thông dụng:

- sort - sắp xếp (mặc định theo thứ tự tăng dần)
- ceil - làm tròn số lên
- floor - làm tròn số xuống
- memset - thường dùng để reset bộ nhớ cho mảng
- max_element - tìm phần tử lớn nhất
- next_permutation - sinh hoán vị tiếp theo