Thuật toán ứng dụng

Giảng viên : Đỗ Tuấn Anh, Lê Quốc Trung
TA: Trần Thanh Tùng

Viện Công Nghệ Thông Tin và Truyền Thông
Trường Đại học Bách Khoa Hà Nội

Tháng 4, năm 2020

Mục lục

PIE

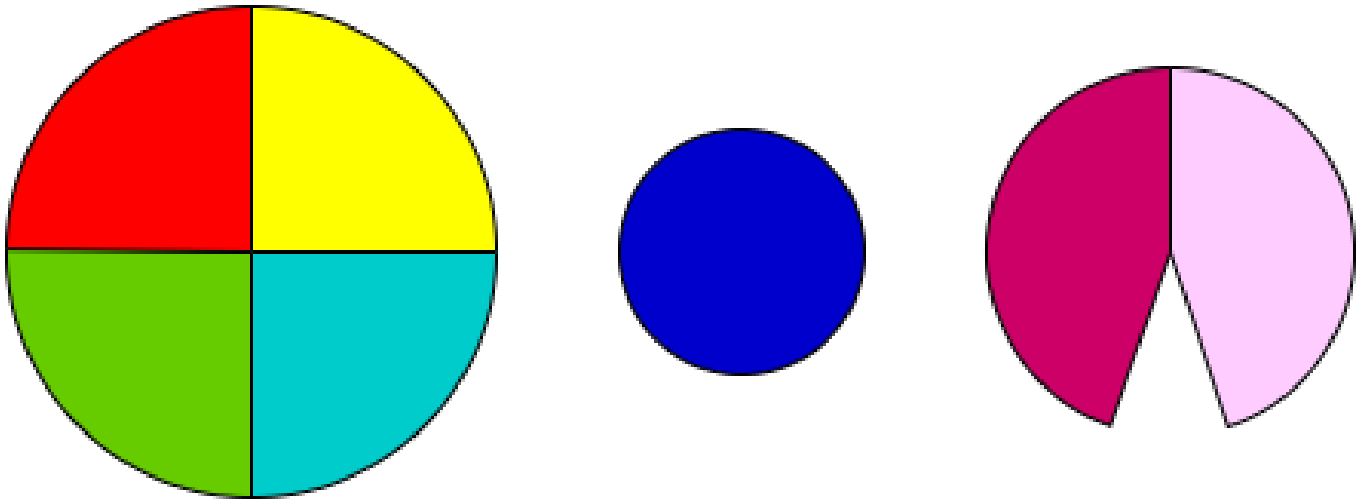
- Lớp bài toán phân chia công bằng
- Vấn đề đặt ra là **nguồn tài nguyên không đồng nhất**
- Ví dụ : Đối với nguồn tài nguyên không phân chia được
- Tài sản thừa kế
- Phân bổ dụng cụ
- Phân bổ nguồn lực
- Đối với Những tài nguyên có thể cắt được
- Bài toán chia bánh

Phát biểu bài toán

- Có N cái bánh và $F + 1$ người.
- Bánh thứ i có bán kính r_i và chiều cao là 1.
- Mỗi người chỉ được nhận một miếng bánh từ một chiếc bánh.
- Cần chia bánh sao cho mọi người có lượng bánh bằng nhau (có thể bỏ qua vụn bánh).
- Tìm lượng bánh lớn nhất mỗi người nhận được.

Ví dụ

- Giả sử có 3 cái bánh có bán kính lần lượt là 2; 1; 1.5
- Cần chia 3 cái bánh trên cho 7 người
- Cách chia sau là tối ưu, mỗi người nhận được một phần là π



Thuật toán

- Gọi $p[i]$ là số người ăn chiếc bánh thứ i. Lượng bánh mỗi người nhận được là $min_i\{\frac{V[i]}{p[i]}\}$ với $V[i]$ là thể tích của chiếc bánh thứ i.
- **Cách 1 - Tìm kiếm theo mảng p :** Tìm kiếm vét cạn mọi giá trị của p.
- **Cách 2 - Tìm kiếm theo lượng bánh mỗi người nhận được :** Thử từng kết quả, với mỗi kết quả, kiểm tra xem có thể chia bánh cho tối đa bao nhiêu người.
- **Tối ưu cách 2 :** Sử dụng thuật toán tìm kiếm nhị phân để tìm kiếm kết quả.

Code

```
1     sort(r, r + N);
2
3     double lo = 0, hi = 4e8, mi;
4
5     for(int it = 0; it < 100; it++){
6         mi = (lo + hi) / 2;
7
8         int cont = 0;
9
10        for(int i = N - 1;i >= 0 && cont <= F; --i)
11            cont += (int)
12            floor(PI * r[i] * r[i] / mi);
13
14        if(cont > F) lo = mi;
15        else hi = mi;
16    }
17    printf("%0.6f", low);
```

EKO

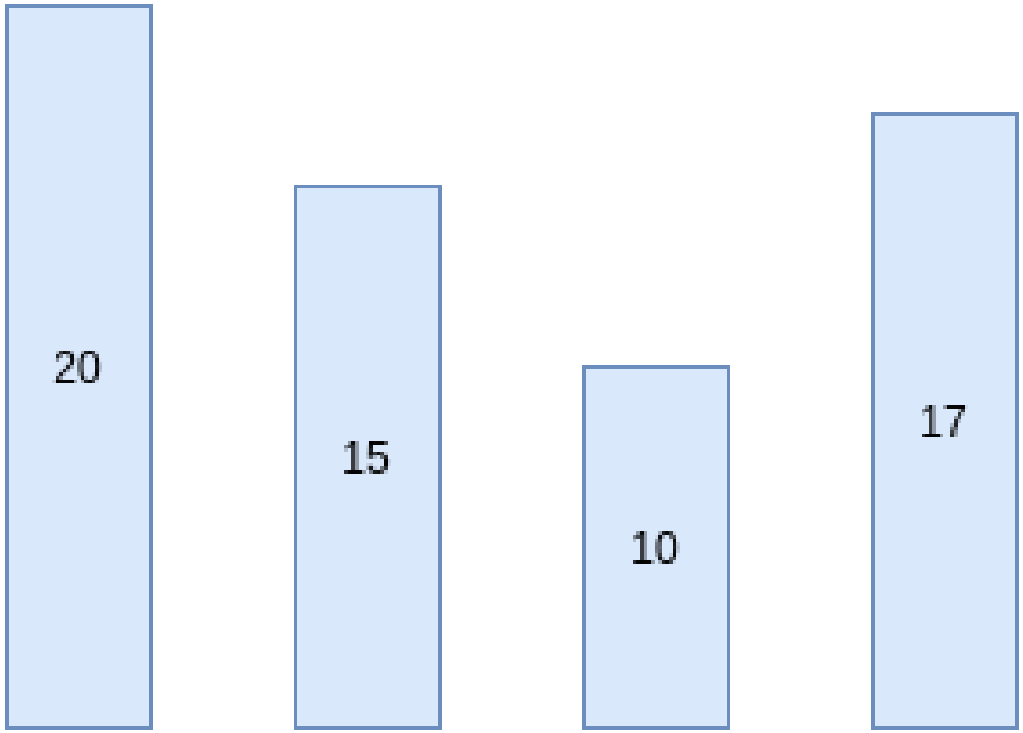
- Cho n cái cây có chiều cao khác nhau $a_1, a_2, ...a_n$
- Có thể thực hiện một phát cắt độ cao h với tất cả các cây.
- Số lượng gỗ thu được là phần chóp của các cây cao hơn h.
- Tìm h lớn nhất có thể để số lượng gỗ thu được lớn hơn m.

Phát biểu bài toán

- Cho n cái cây có chiều cao khác nhau $a_1, a_2, ...a_n$
- Có thể thực hiện một phát cắt độ cao h với tất cả các cây.
- Số lượng gỗ thu được là phần chóp của các cây cao hơn h.
- Tìm h lớn nhất có thể để số lượng gỗ thu được lớn hơn m.

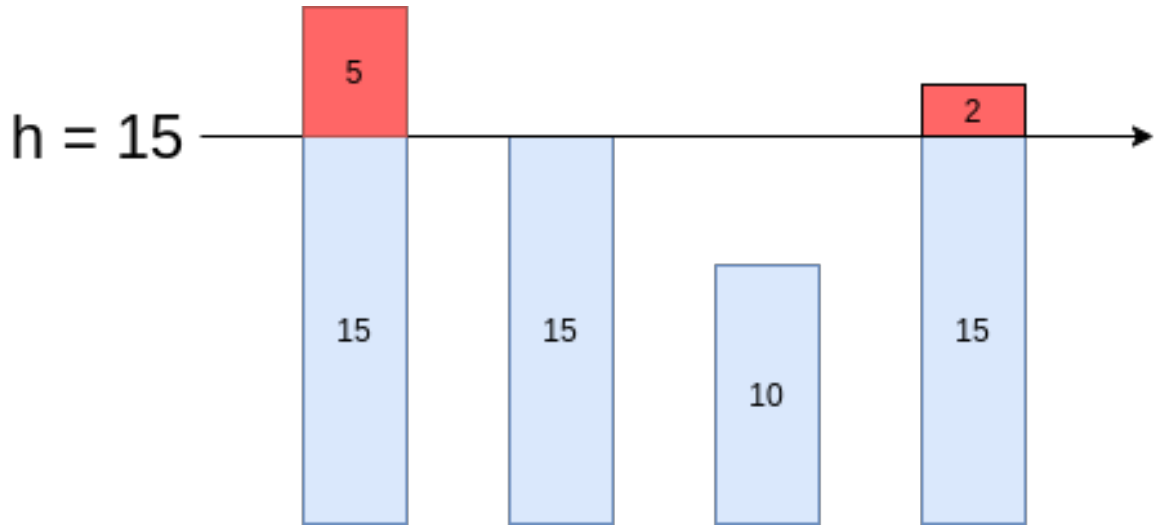
Ví dụ

- Có 4 cây $20, 15, 10, 17$, lượng gỗ tối thiểu cần cắt là 7.



Ví dụ

- Chọn $h = 15 \rightarrow$ số lượng gỗ thu được ở mỗi cây là $5, 0, 0, 2$. tổng là 7.
- Vậy chiều cao lớn nhất có thể cắt được là 15



Thuật toán

- **Thuật toán 1 :** tìm tất cả các giá trị $h \in \{0, max(a[i])\}$. Với mỗi h, tính số lượng gỗ thu được. ĐPT : $O(max(a[i]) * n)$.
- **Thuật toán 2 :** chặt nhị phân giá trị h.

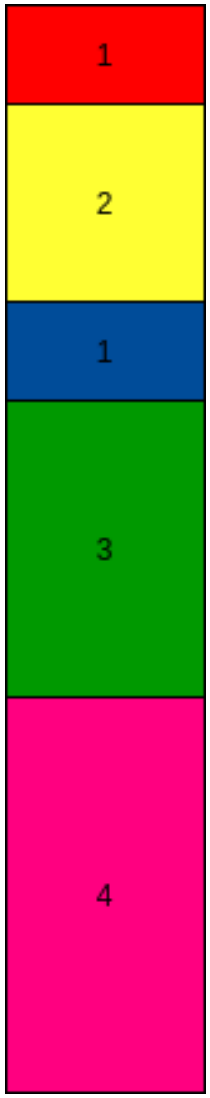
Code

```
18 long long count_wood(int height) {
19     long long sum = 0;
20     for (int i = 1; i <= n; i++)
21         if ( a[i] > height )
22             sum += a[i] - height;
23     return sum;
24 }
25
26 int main {
27     int l = 0, r = max(r,a[i]);
28
29     while (r - l > 1) {
30         int mid = (l + r) / 2;
31         if (count_wood(mid) >= m ) l = mid;
32         else r = mid;
33     }
34     cout << l;
35 }
```

BOOKS1

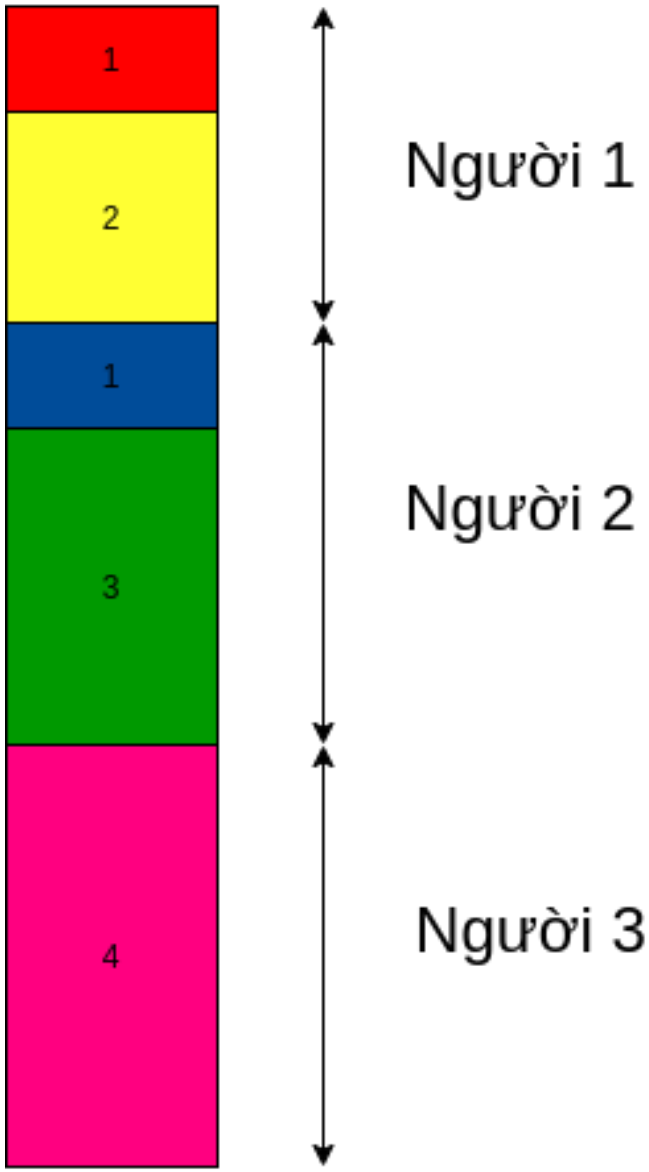
- Có m quyển sách, quyển sách thứ i dày p_i trang.
- Phải chia số sách trên cho đúng k người, mỗi người sẽ nhận được một đoạn sách liên tiếp nhau.
- In ra cách chia để số trang sách lớn nhất được nhận bởi một người là nhỏ nhất.
- Nếu có nhiều kết quả lớn nhất thì ưu tiên số sách nhận bởi người 1 là ít nhất, sau đó đến người 2, ...

Ví dụ



- Đầu vào có 5 quyển sách và phải chia số sách trên cho 3 người
- Mỗi quyển sách có độ dày như hình bên

Ví dụ



- Kết quả của bài toán là 4
- Có 2 cách chia để đạt được kết quả trên :
  1 2/1 3/4 hoặc
  1 2 1/3/4
- Cách chia như hình bên là kết quả của bài toán

Thuật toán 1

- Duyệt kết quả của bài toán từ nhỏ đến lớn, cố định số trang sách lớn nhất được chia cho 1 người.
- Với mỗi kết quả ta đi kiểm tra xem có thể chia được cho đúng k người hay không bằng thuật toán tham lam.
- In ra kết quả ngay khi tìm được kết quả thỏa mãn
- Độ phức tạp thuật toán $O(MAX * n)$

Code 1

```
36 bool check(long long max_val) {
37     vector < int > pos;
38     long long sum = 0;
39     for (int i = n; i >= 1; i--) {
40         if (sum + a[i] <= max_val) {
41             sum += a[i];
42         } else {
43             sum = a[i];
44             if (a[i] > max_val) { return false; }
45             pos.push_back(i); // cac phan tu ngan cach
46         }
47     }
48     if (pos.size() >= k) { return false; }
49     ** In ket qua **
50     return true;
51 }
```

Thuật toán 2

- Gọi *maxVal* là số trang lớn nhất được chia bởi 1 người.
- Nhận thấy nếu với giá trị $maxVal = x$ có thể chia dãy thành $\leq k$ đoạn thì với $maxVal = x + 1$ cũng có thể chia dãy thành $\leq k$ đoạn với cách chia như cũ.
- Ta chặt nhị phân giá trị *maxVal*.
- Độ phức tạp thuật toán $O(\log MAX * n)$

Code 2

```
52 bool check(long long max_val) {
53     // Giong voi ham o Code 1
54 }
55 int main() {
56     int q; cin >> q;
57     while (q--) {
58         cin >> n >> k;
59         for (int i = 1; i <= n; i++) { cin >> a[i]; }
60         long long l = 0, r = MAX;
61         while (r - l > 1) {
62             long long mid = (l + r) >> 1;
63             if (check(mid)) {
64                 r = mid;
65             } else {
66                 l = mid;
67             }
68         }
69         ** In kq tuong ung voi gia tri r **
70     }
71 }
```