Thuật toán ứng dụng
Bài thực hành số 3: Chia để trị

TS. Nguyễn Tuấn Dũng, ThS. Nguyễn Duy Hiệp, ThS. Tạ Minh Trí,
TA. Đặng Xuân Vương

Trường Đại học Bách khoa Hà Nội
Viện Công nghệ thông tin và Truyền thông

Ngày 15 tháng 4 năm 2020

Mục lục

Mục lục

04. PIE (VUONGDX)

- Có N cái bánh và $F + 1$ người.
- Mỗi cái bánh có hình tròn, bán kính r 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.

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.
  - Cho một số x bất kỳ, nếu có thể chia bánh sao cho mỗi người có lượng bánh là x thì cũng có thể chia bánh sao cho mỗi người có lượng bánh là $x', \forall x' < x$.
  - Tương tự, nếu không thể chia bánh sao cho mỗi người có lượng bánh là x thì cũng không thể chia bánh sao cho mỗi người có lượng bánh là $x', \forall x' > x$.

Cài đặt

- **Điều kiện dừng của tìm kiếm nhị phân thông thường (trên miền số nguyên) không xảy ra:**
  - Tìm kiếm trên miền số thực nên không có giá trị nhỏ nhất lớn hơn một giá trị cho trước (hoặc giá trị lớn nhất nhỏ hơn một giá trị cho trước).
  - *low* và *high* không bao giờ bằng nhau.
- **Sai số khoảng giảm dần:**
  - Mỗi lần tìm kiếm nhị phân làm khoảng giá trị giảm đi 2 lần.
  - Kết quả tìm được có sai số không vượt quá khoảng tìm kiếm chứa nó.
  - Cần giảm khoảng giá trị $(high - low)$ xuống 10^{-6}.
  - Dùng 100 bước lặp có thể giảm khoảng giá trị ban đầu đi 10^{30} lần.
- **Lưu ý khi sử dụng hằng số PI:** Sử dụng hằng số của thư viện hoặc dùng giá trị đủ chính xác (Ví dụ: 3.14159265358979323846264338327950288).
- **Sắp xếp bánh theo kích thước:** Khi kiểm tra, duyệt từ chiếc bánh to nhất trước giúp tăng cơ hội kết thúc hàm kiểm tra sớm hơn.

Code

```
// r[i]: binh phuong ban kinh cua moi chiec banh
sort(r, r + N);

    double lo = 0, hi = M_PI * max(r), mi;

    for(int it = 0; it < 100; it++){
        mi = (lo + hi) / 2;

        int cont = 0;

        for(int i = N - 1;
        i >= 0 && cont <= F; --i)
            cont += (int)
            floor(M_PI * r[i] / mi);

        if(cont > F) lo = mi;
        else hi = mi;
    }
```

Mục lục

04. FIBWORDS (vuongdx)

- Dãy Fibonacci Words của xâu nhị phân được định nghĩa như sau:

$$F(n) = \begin{cases} 0, & \text{if } n = 0 \\ 1, & \text{if } n = 1 \\ F(n-1) + F(n-2), & \text{if } n \geq 2 \end{cases}$$

- Cho n và một xâu nhị phân p. Đếm số lần p xuất hiện trong $F(n)$ (các lần xuất hiện này có thể chồng lên nhau).
- Giới hạn: $0 \leq n \leq 100$, p có không quá 100000 ký tự, kết quả không vượt quá 2^{63}.

Thuật toán I

- **Thuật toán 1 - Vét cạn:** So sánh xâu p với mọi xâu $f(n)[i..(i + len(p)]$.
- **Thuật toán 2 - Chia để trị:** Xâu $f(n)$ gồm 2 xâu con là $f(n-1)$ và $f(n-2)$.
    - Đếm số lần p xuất hiện trong $f(n-1)$, $f(n-2)$.
    - Đếm số lần p xuất hiện ở đoạn giữa của xâu $f(n)$ (đoạn đầu của p là đoạn cuối của $f(n-1)$, đoạn cuối của p là đoạn đầu của $f(n-2)$).

Thuật toán II

- Đếm số lần p xuất hiện trong $f(i)$ với i nhỏ: Sử dụng **thuật toán 1**.
- Đếm số lần p xuất hiện ở đoạn giữa xâu $f(n)$:
  - Giả sử 2 xâu $f(i-1)$ và $f(i)$ có độ dài lớn hơn độ dài xâu p, $f(i-1)$ có dạng $x..a$, $f(i)$ có dạng $y..b$, trong đó x, y, a, b có độ dài bằng độ dài của p (x và a hay y và b có thể chồng lên nhau).
  - **Nhận xét 1:** $x = y$.
  - **Nhận xét 2:** Nếu $n \equiv i(mod 2)$ thì đoạn giữa của $f(n)$ là $..ax..$, ngược lại, đoạn giữa của $f(n)$ là $..bx..$.

Thuật toán III

- Cài đặt:
  - **void preprocessing()**: Tính trước các xâu fibonacci word, 2 xâu cuối cùng có độ dài không nhỏ hơn 10^5.
  - **long long count(string s, string p)**: Đếm số lần p xuất hiện trong s theo thuật toán 1.
  - **long long count(int n, string p)**: Đếm số lần p xuất hiện trong $f(n)$ theo thuật toán 2.
  - **long long solve(int n, string p)**:
    - Xử lý trường hợp $f(n)$ có độ dài nhỏ hơn độ dài của p.
    - Khởi tạo mảng c - $c[i]$ là số lần xuất hiện của p trong $f(i)$.
    - Sử dụng hàm count(s, p) để đếm số lần xuất hiện của p trong $f(i)$ và $f(i-1)$ với $f(i-1)$ là fibonacci word đầu tiên có độ dài không nhỏ hơn độ dài của p rồi lưu vào mảng c.
    - Sử dụng hàm count(s, p) để đếm số lần xuất hiện của p trong ax và bx, lưu vào mảng mc.
    - Sử dụng hàm count(n, p) để đếm số lần xuất hiện của p trong $f(n)$.

Code

```
long long solve(int n, string p) {
    int lp = p.size();
    if (n < n_prepare && l[n] < lp) {return 0;}
    for (int j = 0; j <= n; j++) {c[j] = -1;}
    int i = 1;
    while (l[i - 1] < lp) {i++;}
    c[i - 1] = count(f[i - 1], p);
    c[i] = count(f[i], p);
    string x = f[i].substr(0, lp - 1);
    string a =
    f[i - 1].substr(f[i - 1].size() - (lp - 1));
    string b =
    f[i].substr(f[i].size() - (lp - 1));
    mc[i % 2] = count(a + x, p);
    mc[(i + 1) % 2] = count(b + x, p);
    return count(n, p);
}
```

Code

```
long long count(int n, string p) {
    if (c[n] < 0) {
        c[n] = count(n - 1, p)
        + count(n - 2, p)
        + mc[n % 2];
    }
    return c[n];
}
```

Thuật toán ứng dụng
Bài thực hành số 3: Chia để trị

TS. Nguyễn Tuấn Dũng, ThS. Nguyễn Duy Hiệp, ThS. Tạ Minh Trí,
TA. Đặng Xuân Vương

Trường Đại học Bách khoa Hà Nội
Viện Công nghệ thông tin và Truyền thông

Ngày 15 tháng 4 năm 2020