Weekly Report - Midterm Test

Hoàng Anh

Ngày 22 tháng 7 năm 2025

Bài toán 1: Ferrers Diagram

Mục đích

Chương trình này sinh ra tất cả các phân hoạch của một số nguyên n thành k phần, và với mỗi phân hoạch, vẽ biểu đồ Ferrers và biểu đồ Ferrers chuyển vị (transpose). Có hai phiên bản: một bằng C++ và một bằng Python.

Giải thích thuật toán

- **Sinh phân hoạch:** Sử dụng đệ quy để sinh tất cả các phân hoạch của n thành k số nguyên dương, không tăng dần.
- **Vẽ Ferrers diagram:** Với mỗi phân hoạch, in ra mỗi dòng số lượng dấu * tương ứng với từng phần tử, căn phải theo phần tử lớn nhất.
- **Vẽ Ferrers transpose:** In ra ma trận chuyển vị của Ferrers diagram, tức là hoán đổi hàng và cột.

Phân tích chi tiết cách hoạt động của thuật toán

Thuật toán trong cả hai chương trình (C++ và Python) gồm hai phần chính: sinh phân hoạch số và vẽ biểu đồ Ferrers (cùng chuyển vị). Dưới đây là phân tích chi tiết:

1. **Sinh phân hoạch số:**
    - Ý tưởng là dùng đệ quy để liệt kê tất cả các cách phân tích số n thành k số nguyên dương, mỗi số không lớn hơn một giá trị cho trước (ban đầu là n).
    - Ở mỗi bước đệ quy, thuật toán chọn một số i (từ $\min(n, \text{max\_val})$ đến 1), thêm vào phân hoạch hiện tại, rồi tiếp tục phân tích $n-i$ thành $k-1$ phần với số lớn nhất là i.

- Khi $k = 0$ và $n = 0$, một phân hoạch hợp lệ được tạo ra và sẽ được xử lý tiếp.

2. **Vẽ Ferrers diagram:**

   - Với mỗi phân hoạch, biểu đồ Ferrers được vẽ bằng cách in ra mỗi dòng số lượng dấu * tương ứng với từng phần tử của phân hoạch, căn phải theo phần tử lớn nhất.
   - Ví dụ, phân hoạch $(3, 1)$ sẽ vẽ:

     ```
     *** 3
     *   1
     ```

3. **Vẽ Ferrers transpose:**

   - Biểu đồ Ferrers chuyển vị (transpose) là hoán đổi hàng và cột của biểu đồ gốc.
   - Cách làm là duyệt từng cột (theo chiều cao lớn nhất của phân hoạch), với mỗi cột kiểm tra xem từng hàng có dấu * không (tức là phần tử phân hoạch lớn hơn chỉ số cột).
   - Ví dụ, transpose của $(3, 1)$ là:

     ```
     * *
     *
     *
     3 1
     ```

Giải thích mã nguồn C++

- `maxp(const vector<int>& p)`: Tìm phần tử lớn nhất trong phân hoạch p.
- `Ferrers(const vector<int>& p)`: In biểu đồ Ferrers cho phân hoạch p.
- `FerrersTrans(const vector<int>& p)`: In biểu đồ Ferrers chuyển vị cho phân hoạch p.
- `genF(int n, int k, vector<int>& cur, int max_val)`: Đệ quy sinh các phân hoạch của n thành k phần, mỗi phần không lớn hơn max_val.
- `main()`: Đọc n, k từ đầu vào, gọi hàm sinh phân hoạch và in kết quả.

Giải thích mã nguồn Python

- `max_part(partition)`: Tìm phần tử lớn nhất trong phân hoạch.
- `print_ferrers(partition)`: In biểu đồ Ferrers.
- `print_ferrers_transpose(partition)`: In biểu đồ Ferrers chuyển vị.
- `generate_partitions(n, k, current, max_val)`: Đệ quy sinh phân hoạch.
- `main()`: Đọc n, k từ đầu vào, gọi hàm sinh phân hoạch và in kết quả.

Ví dụ đầu vào/đầu ra

```
Input:
4 2

Output:
F:
** 2
** 2
FT:
* *
* *
2 2
====================
F:
*** 3
*   1
FT:
* *
*
*
3 1
====================
```

So sánh hai phiên bản

- Cả hai phiên bản đều sử dụng cùng một thuật toán sinh phân hoạch và in biểu đồ Ferrers.
- Phiên bản C++ sử dụng `vector<int>` và thao tác nhập/xuất chuẩn của C++.

- Phiên bản Python sử dụng list và cú pháp Pythonic, dễ đọc hơn.
- Kết quả đầu ra của hai chương trình là tương đương.

Mã nguồn C++

```
#include <bits/stdc++.h>
using namespace std;

int maxp(const vector<int>&p) {
    int pmax = 0;
    int n = p.size();
    for (int i = 0; i < n; ++i) {
        if (p[i] > pmax)
            pmax = p[i];
    }
    return pmax;
}

void Ferrers(const vector<int>&p) {
    cout << "F:\n";
    int pmax = maxp(p);
    for (int r : p) {
        for (int i = 0; i < r; ++i) cout << '*';
        for (int i = 0; i < pmax - r; ++i) cout << "␣";
        cout << "␣" << r << '\n';
    }
}

void FerrersTrans(const vector<int>& p) {
    cout << "FT:\n";
    int pmax = maxp(p);
    int n = p.size();
    for (int r = 0; r < pmax; ++r) {
        for (int i = 0; i < n; ++i) {
            if (p[i] > r) cout << "*␣";
            else cout << "␣";
        }
        cout << '\n';
    }
    for (int i = 0; i < n; ++i) {
        cout << p[i];
        if (i != n - 1) cout << "␣";
    }
    cout << "\n";
}
```

```
void genF(int n, int k, vector<int>&cur, int max_val) {
    if (k == 0) {
        if (n == 0) {
            Ferrers(cur);
            FerrersTrans(cur);
            cout << "====================\n";
        }
        return;
    }
    for (int i = min(n, max_val); i >= 1; --i) {
        cur.push_back(i);
        genF(n - i, k - 1, cur, i);
        cur.pop_back();
    }
}

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    int n, k;
    cin >> n >> k;
    vector<int> current;
    genF(n, k, current, n);
    return 0;
}
```

Mã nguồn Python

```
# ferreries_diagram.py

def max_part(partition):
    return max(partition) if partition else 0

def print_ferrers(partition):
    print("F:")
    pmax = max_part(partition)
    for r in partition:
        print("*" * r + "␣" * (pmax - r) + f"␣{r}")

def print_ferrers_transpose(partition):
    print("FT:")
    pmax = max_part(partition)
    n = len(partition)
    for r in range(pmax):
        row = ''
        for i in range(n):
```

```
            row += "*␣" if partition[i] > r else "␣␣"
        print(row.rstrip())
    print('␣'.join(str(x) for x in partition))

def generate_partitions(n, k, current, max_val):
    if k == 0:
        if n == 0:
            print_ferrers(current)
            print_ferrers_transpose(current)
            print("====================")
        return
    for i in range(min(n, max_val), 0, -1):
        current.append(i)
        generate_partitions(n - i, k - 1, current, i)
        current.pop()

def main():
    n, k = map(int, input().split())
    generate_partitions(n, k, [], n)

if __name__ == "__main__":
    main()
```