BÀI THỰC HÀNH SỐ 6: CHIẾN LƯỢC QUY HOẠCH ĐỘNG

Chiến lược quy hoạch động

- Quy hoạch động là một chiến lược nhằm đơn giản hóa việc tính toán các công thức truy hồi bằng cách lưu trữ toàn bộ hay một phần kết quả tính toán tại mỗi bước với mục đích sử dụng lại.
- Bản chất của quy hoạch động là thay thế mô hình tính toán top – down bằng mô hình tính toán bottom – up.
- Quy hoạch động giúp giải quyết các bài toán tối ưu mang bản chất đệ quy.
- Đặc điểm chung của quy hoạch động:
    - Quy hoạch động bắt đầu từ việc giải tất cả các bài toán nhỏ nhất (bài toán cơ sở) để từ đó từng bước giải quyết những bài toán lớn hơn cho tới khi giải được bài toán lớn nhất (bài toán ban đầu).
    - Quy hoạch động cần phải có bảng phương án.
    - Ý tưởng cơ bản của phương pháp quy hoạch động là tránh tính toán lại các bài toán con đã xét.

Bài tập 1: Phân tích số thành tổng

Input: Cho số tự nhiên n $\leq$100.

Output: Tính p là số cách phân tích số n thành tổng của dãy các số nguyên dương.

Yêu cầu thực hiện:

- Thiết kế thuật toán Q1 để tính và trả về p bằng chiến lược quy hoạch động.
- Cài đặt chương trình ứng dụng:
    - Cài đặt hàm biểu diễn thuật toán Q1.
    - Khởi tạo n.
    - Sử dụng thuật toán Q1 để tính và đưa ra p.

Bài tập 2: Bài toán cái túi

Input:

- Cho danh sách n gói hàng được đánh số thứ tự từ 1 đến n, các gói hàng có trọng lượng lần lượt là w_1, w_2, ..., w_n và giá trị lần lượt là v_1, v_2, ..., v_n.
- Cho một chiếc túi có thể chứa được khối lượng tối đa là m.

Output:

- Tính u và d lần lượt là số gói hàng và danh sách các gói hàng cần xếp vào cái túi sao cho giá trị lấy được là lớn nhất mà không vượt quá kích thước của cái túi.

Yêu cầu thực hiện:

- Thiết kế thuật toán Q2 để tính và trả về u và d bằng chiến lược quy hoạch động.
- Cài đặt chương trình ứng dụng:
    - Cài đặt hàm biểu diễn thuật toán Q2.
    - Khởi tạo m, n, $\{w_1, w_2, ..., w_n\}$và $\{v_1, v_2, ..., v_n\}$.
    - Sử dụng thuật toán Q2 để tính và đưa ra u và d.

Bài tập 3: Dãy con đơn điệu tăng dài nhất.

Input:

- Cho số tự nhiên n và dãy a gồm n số thực $a_1, a_2, ..., a_n$.
- Một dãy con của dãy a là một cách chọn ra trong dãy a một số phần tử giữ nguyên thứ tự (dãy a có 2^n dãy con như vậy).

Output:

- Tính b là một dãy con của dãy a với b gồm các phần tử có thứ tự tăng với số phần tử là nhiều nhất (dãy b được gọi là dãy con đơn điệu tăng dài nhất).

Yêu cầu thực hiện:

- Thiết kế thuật toán Q3 để tính và trả về b bằng chiến lược quy hoạch động.
- Cài đặt chương trình ứng dụng:
    - Cài đặt hàm biểu diễn thuật toán Q3.
    - Khởi tạo n và dãy a.
    - Sử dụng thuật toán Q3 để tính và đưa ra b.

Bài tập 4: Coin changing.

Input:

- Cho n loại tiền có mệnh giá lần lượt là $c_1, c_2, ..., c_n$ với số lượng tờ tiền mỗi loại mệnh giá không giới hạn.
- Cho một số tiền m.

Output:

- Tính $s = \{s_1, s_2, ..., s_n\}$ là số tờ tiền mỗi loại cần lấy để chi trả vừa đủ số tiền m sao cho tổng số lượng tờ tiền phải trả t là ít nhất.

Yêu cầu thực hiện:

- Thiết kế thuật toán Q4 để tính và trả về s và t bằng chiến lược quy hoạch động.
- Cài đặt chương trình ứng dụng:
    - Cài đặt hàm biểu diễn thuật toán Q4.
    - Khởi tạo n và $\{ c_1, c_2, ..., c_n \}$.
    - Sử dụng thuật toán Q4 để tính và đưa ra s và t.

Hướng dẫn:

- Gọi f(i) là số lượng tờ ít nhất để trả số tiền i nên f(m) là số tờ ít nhất để trả số tiền m.
- Để được số tiền là i có các cách để tạo thành số tiền đó khi chúng ta dùng thêm một tờ là: i - $c[k_1]$, i - $c[k_2]$, ..., i - $c[k_j]$, trong đó k_j là số thoả mãn mà $c[k_j] < i$.
- Số tờ tiền tối ưu nhất cần tìm là giá trị nhỏ nhất trong các giá trị:
  $f(i - c[k_1]) + 1$, $f(i - c[k_2]) + 1$, ..., $f(i - c[k_j]) + 1$.
- Công thức quy hoạch động như sau:

  $f(i) = \min\{f(i - c[j]) + 1$, j thoả mãn: $c[j] < i$

- Thuật toán quy hoạch động:

```
coin_changing() {
    f[0] = 0;
    for (i = 1 -> m) {
        min = maxint;
        for (j = 1 -> n) {
            if (f[i - a[j]] + 1 < min && a[j] < i) {
                min = f[i - a[j]] + 1;
                s[i] = j;
            }
        }
        f[i] = min;
    }
}
```

Giải thích:

s là mảng đánh dấu loại tiền nào cần dùng cuối cùng để đạt số tiền i.

Truy vết để tìm lại các loại tiền cần dùng bằng mảng s như sau:

```
j = s[m];
i = m;
while (j khác 0) {
    print(a[j]);
    i = i - j;
    j = s[i];
}
```