**Tóm tắt đề:** Xoá đi x kí tự từ xâu S tạo ra xâu S’ tìm cách xoá để p xuất hiện nhiều lần trong S’ nhất ( không đè nhau ) với từng x từ 0 → |S|

Gọi N = |S|, M = |P|

**Subtask 1:** N, M <= 20

Sinh nhị phân ra S’ và tìm các lần xuất hiện của P trong S’. Giả sử để tạo ra S’ cần xoá x kí tự và P xuất hiện T trong S’ thì cập nhật F[x] = max{ T }

**Subtask 2:** N, M <= 300

Gọi F[i][cur][rem] có ý nghĩa số lần xuất hiện nhiều nhất của P khi xét tới vị trí i của S và còn rem kí tự cần xoá

Công thức quy hoạch động:

F[i][cur][rem] ← F[i + 1][nxt][rem] + (cur == M)

F[i][cur][rem] ← F[i + 1][cur][rem - 1]

Trong đó : nếu cur = M thì nxt = 1 còn không thì phân ra các trường hợp:

* S[i] != P[cur]  => nxt = 1;
* S[i] == P[cur] => nếu cur < M thì nxt = cur + 1 còn cur == M thì nxt = 1 và tăng đáp án thêm 1

**Subtask 3:** N <= 2000, M <= 500

Nhận thấy điều duy nhất thay đổi được đáp án là một phát ăn cả xâu P và xoá đi những cái không cần thiết.

P = “abc”

S = “a….b…c..xyz” thì cách duy nhất để tăng đáp án là xoá toàn bộ phần … giữa a...b hoặc là không xoá hoặc là xoá sạch thì mới tăng được đáp án. Nhưng nếu P có nhiều hơn 2 phần tử thì phải làm cách nào đấy xoá sạch các đoạn giữa 2 phần tử liên tiếp của P trong S nghĩa là xoá hết … trong “a...b...c”

Nhận xét: Ta chỉ quan tâm chi phí xoá sạch ở vị trí bất kì của S là bao nhiêu để nhận 1 xâu P và vị trí tiếp theo ta nhảy tới sau khi xoá.

Từ đây ta có 1 ý tưởng : với mỗi vị trí i của xâu S ban đầu thì ta chuẩn bị trước mảng eat[i] là ta ăn sạch ở vị trí i thì tốn bao nhiêu lần xoá. Cách xây dựng mảng eat: cứ tham lam mà đánh. Và kèm với mảng eat là mảng go[i] là vị trí ta sẽ nhảy tới khi xoá sạch để lấy xâu P

FOR(i, 0, n - 1){

       ll pos = 0;

       ll cost = 0;

       go[i] = -1;

       FOR(j, i, n - 1){

           if (s[j] == p[pos])

               pos++;

           else cost++;

           if (pos >= m) {go[i] = j;break;}

       }

       eat[i] = cost;

   }

Gọi F[i][re] là số lần xuất hiện nhiều nhất của xâu P trong S khi xét tới vị trí i và còn re lần xoá

Công thức quy hoạch động:

F[i][re] ← F[i + 1][re]

F[i][re] ← F[go[i] + 1][re - eat[i]] + 1 với điều kiện: go[i] != -1 và re - eat[i] >= 0