Đề bài yêu cầu tìm **hoán vị phân biệt thứ k nhỏ nhất** theo thứ tự từ điển của một mảng có độ dài n, sao cho giá trị **F(P)** đạt giá trị lớn nhất.

**Các khái niệm:**

* Một mảng A là hoán vị nếu nó chứa n số nguyên khác nhau từ 1 đến n theo thứ tự bất kỳ.
* F(P) là số lượng dãy con liên tiếp trong P là hoán vị. Ví dụ:
  + Với P bằng (5, 3, 1, 4, 2), các dãy con liên tiếp là hoán vị:
    - (1)
    - (3, 1, 4, 2)
    - (5, 3, 1, 4, 2).
  + Vậy F(P) bằng 3.
* Một hoán vị P được gọi là **phân biệt** nếu F(P) đạt giá trị lớn nhất trong tất cả các hoán vị độ dài n.
* Nhiệm vụ: Với n và k cho trước, tìm **hoán vị phân biệt thứ k** theo thứ tự từ điển.

**Input:**

* t là số lượng bộ test.
* Với mỗi bộ test, gồm hai số nguyên n và k.

**Output:**

* Với mỗi bộ test, xuất ra hoán vị phân biệt thứ k nhỏ nhất theo thứ tự từ điển.

**Ràng buộc:**

* 1 <= t <= 1e5
* 1 <= n <= 1e5
* 1 <= k <= 1e18

**Ví dụ:**

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| 1  4 2 | 2 1 3 4 |

**Lời giải:**

Biết rằng thứ tự từ điển được sắp xếp theo : 1 < 2 < 3 < 4 < 5 < 6….

Gọi g(n) là số lượng hoán vị có số n đứng đầu và được sắp xếp theo thứ tự từ điển

Thử nháp ra các trường hợp hoán vị sắp xếp theo thứ tự từ điển khi n = 5:

**Khi 1 đứng đầu: g(1) = 1**

1 2 3 4 5

**Khi 2 đứng đầu: g(2) = 1**

2 1 3 4 5

**Khi 3 đứng đầu: g(3) = 2**

3 1 2 4 5

3 2 1 4 5

**Khi 4 đứng đầu: g(4) = 4**

4 1 2 3 5

4 2 1 3 5

4 3 1 2 5

4 3 2 1 5

**Ta nhận thấy rằng** g(1) = 1, g(2) = 1 **và khi n > 2 thì độ g(n) sẽ bằng hay** g(n) = 2n-2 (n >= 2)

**Suy ra:** g(5) = g(1) + g(2) + g(3) + g(4) = 8 = 23 (\*)

**Chứng minh bằng phương pháp quy nạp:**

Khi 5 đứng đầu:

5 1 2 3 4

5 2 1 3 4

5 3 1 2 4

5 3 2 1 4

5 4 1 2 3

5 4 2 1 3

5 4 3 1 2

5 4 3 2 1

**=>** Có 8 hoán vị, thỏa mãn (\*)

Xét thêm mảng cộng dồn f(n) = g(n) + f(n-1), hay f(n) = 2n-1 với n >= 1

**Ta rút ra 1 bảng:**

|  |  |  |
| --- | --- | --- |
| **n** | **g(n)** | **f(n)** |
| 1 | 1 | 1 |
| 2 | 1 | 2 |
| 3 | 2 | 4 |
| 4 | 4 | 8 |
| 5 | 8 | 16 |
| … | … |  |

Ta nhận xét rằng, hoán vị i có độ dài n và bé thứ k sẽ nằm trong block f(i) với f(i) >= k > f(i-1)

Xét lại test case mẫu:

|  |  |
| --- | --- |
| **INPUT** | **OUTPUT** |
| 1  4 2 | 2 1 3 4 |

Xét các hoán vị có độ dài 4, hoán vị bé thứ 2 sẽ nằm trong block thứ 2 vì f(1) = 1 < 2 <= f(2) = 2

=> 2 1 3 4 (đúng)

Xét khi: n = 4, k = 3, hoán vị bé thứ 3 sẽ nằm trong block thứ 3 vì f(2) = 2 < 3 <= f(3) = 4

Thì hoán vị thỏa mãn sẽ là 3 1 2 4 (đúng)

**Từ những nhận xét trên, rút ra ý tưởng bài toán:**

* Để f(P) đạt giá trị lớn nhất, hoán vị cần có thứ tự các phần tử sao cho số lượng dãy con liên tiếp là hoán vị được tối ưu. Điều này đạt được khi các phần tử lớn hơn xuất hiện càng gần cuối mảng.
* Hoán vị thứ k sẽ thuộc block của phần tử i đầu tiên sao cho: f(i-1) < k <= f(i)

**Thuật toán:**

1. **Xác định phần tử đầu tiên:**
   * Sử dụng tìm kiếm nhị phân trên các giá trị f(i) để tìm i thỏa mãn f(i-1) < k <= f(i).
   * Phần tử đầu tiên của hoán vị là i, đồng thời: k = k - f(i-1)
2. **Xử lý các phần tử còn lại:**
   * Loại bỏ i khỏi tập số còn lại.
   * Tiếp tục tìm các phần tử kế tiếp bằng cách áp dụng quy trình tương tự cho n-1 phần tử còn lại với giá trị k đã cập nhật.
3. **Sắp xếp các số chưa được chọn trong quá trình tìm kiếm:**
   * Khi k = 1, các số còn lại được sắp xếp tăng dần theo thứ tự từ điển.

**Code:**

|  |
| --- |
| #include <bits/stdc++.h>  #define int long long  using namespace std;  signed main() {  vector<int>f(64, 0);  f[1] = 1;  for (int i = 2; i <= 63; i++) {  f[i] = pow(2, i - 1);  }  int t; cin >> t;  while (t--) {  int n, k;  cin >> n >> k;  vector<int> a;  vector<int> check(n + 1, false);  while (k > 1) {  int ans = -1;  int l = 1, r = 63;  while (l <= r) {  int mid = (l + r) / 2;  int value = f[mid];  if (value < k) {  l = mid + 1;  }  else {  ans = mid;  r = mid - 1;  }  }  a.push\_back(ans);  check[ans] = true;  k -= f[ans - 1];  }  for (int i = 1; i <= n; i++) {  if (check[i] == false) {  a.push\_back(i);  }  }  for (int i = 0; i < n; i++) {  cout << a[i] << ' ';  }  cout << endl;  }  return 0;  } |