2.5. Thuật toán tham lam (Greedy Algorithm)

2.5.1. Giới thiệu thuật toán

Thuật toán tham lam (Greedy Algorithm): là thuật toán xem xét quá trình giải quyết bài toán bằng việc tạo nên lựa chọn tối ưu cục bộ tại mỗi bước thực hiện với mong muốn tìm ra lựa chọn tối ưu toàn cục. Giải thuật tham lam thường không mang tính tổng quát. Tuy vậy, bằng việc xem xét các lựa chọn tối ưu cục bộ cho ta lời giải gần đúng với phương án tối ưu toàn cục cũng là giải pháp tốt trong thời gian chấp nhận được.

Thuật toán Greedy bao gồm 5 thành phần chính:

1. Một tập các ứng viên mà giải pháp thực hiện tạo ra.
2. Một hàm lựa chọn (selection fuction) để chọn ứng viên tốt nhất cho giải pháp.
3. Một hàm thực thi (feasibility function) được sử dụng để xác định các ứng viên có thể có đóng góp cho giải pháp thực thi.
4. Một hàm mục tiêu (objective function) dùng để xác định giá trị của phương pháp hoặc một phần của giải pháp.
5. Một hàm giải pháp (solution function) dùng để xác định khi nào giải pháp kết thúc.

Ví dụ Thuật toán sắp xếp kiểu chọn (Selection Sort):

Input:

- Dãy số A[] gồm n phần tử

Output:

- Đưa ra dãy được sắp

Thực hiện:

```
Begin:
    For( int  i=0; i<n-1; i++) {//duyệt từ phần tử thứ 0..n-2
        Min-idx =i; //thiết lập vị trí phần tử nhỏ nhất lúc đầu là i
        For( j = i+1; j<n; j++) { //duyệt từ phần tử tiếp theo
            If(A[j]<A[Min-idx) //nếu có phần tử nhỏ hơn
                Min-idx = j; //xác nhận vị trí phần tử nhỏ nhất
        }
        Swap(A[i], A[Min-idx]); //đổi chỗ hai phần tử
    }
End.
```

2.5.2. Activities Selction Problem.

Lựa chọn hành động là bài toán tối ưu tổ hợp điển hình quan tâm đến việc lựa chọn các hành động không mâu thuẫn nhau trong các khung thời gian cho trước. Bài toán được phát biểu như sau:

Cho tập gồm n hành động, mỗi hành động được biểu diễn như bộ đôi thời gian bắt đầu si và thời gian kết thúc fi (i=1, 2, .., n). Bài toán đặt ra là hãy lựa chọn nhiều nhất các hành động có thể thực hiện bởi một máy hoặc một cá nhân mà không xảy ra tranh chấp. Giả sử mỗi hành động chỉ thực hiện đơn lẻ tại một thời điểm.

Input:

- Số lượng hành động: 6
- Thời gian bắt đầu Start []= { 1, 3, 0, 5, 8, 5}
- Thời gian kết thúc Finish[]= { 2, 4, 6, 7, 9, 9}

Output: Số lượng lớn nhất các hành động có thể thực hiện bởi một người.

OPT[] = {0, 1, 3, 4 }

Lời giải. Thuật toán tham lam giải quyết bài toán được thực hiện như sau:

Algorithm Greedy-Activities- Selection(N, S[], F[]):

Input:

- N là số lượng hành động (công việc).
- S[] thời gian bắt đầu mỗi hành động.
- F[] thời gian kết thúc mỗi hành động.

Ouput:

- Danh sách thực thi nhiều nhất các hành động.

Actions:

Bước 1 (sắp xếp). Sắp xếp các hành động theo thứ tự tăng dần của thời gian kết thúc.

Bước 2 (Khởi tạo) Lựa chọn hành động đầu tiên làm phương án tối ưu (OPT=1). N = N\ {i};

Bước 3 (Lặp).

```
for each activities j∈N do {
    if ( S[j] >= F[i] ) {
        OPT = OPT∪j; i = j; N = N\{i}
    }
}
```

Bước 4 (Trả lại kết quả).

```
Return (OPT)
```

EndActions.

Ví dụ. Thuật toán Greedy-ActivitiesN-Selection(int N, int S[], int F[]):

Input:

- Số lượng hành động n = 8.
- Thời gian bắt đầu S[] = {1, 3, 0, 5, 8, 5, 9, 14}.
- Thời gian kết thúc F[] = {2, 4, 6, 7, 9, 9, 12, 18}.

Output:

- OPT = { Tập nhiều nhất các hành động có thể thực hiện bởi một máy}.

Phương pháp tiến hành::

Bước	i= ? j =?	(S[j] >=F[i]) ? i=?	OPT=?
			OPT = OPT$\cup${1}.
1	i=1; j=2	(3>=2): Yes; i=2.	OPT = OPT$\cup${2}.
2	i=2; j=3;	(0>=4): No; i=2.	OPT = OPT$\cup\phi$.
3	i=2; j=4;	(5>=4): Yes; i=4.	OPT = OPT$\cup${4}.
4	i=4; j=5;	(8>=7): Yes; i=5.	OPT = OPT$\cup${5}.
5	i=5; j=6;	(5>=9): No; i=5.	OPT = OPT$\cup\phi$.
6	i=5; j=7;	(9>=9): Yes; i=7.	OPT = OPT$\cup${7}.
7	i=7; j=8;	(14>=12): Yes; i=8.	OPT = OPT$\cup${8}.
OPT = { 1, 2, 4, 5, 7, 8}			

2.5.3. Bài toán n-ropes. Cho n dây với chiều dài khác nhau. Ta cần phải nối các dây lại với nhau thành một dây. Chi phí nối hai dây lại với nhau được tính bằng tổng độ dài hai dây. Nhiệm vụ của bài toán là tìm cách nối các dây lại với nhau thành một dây sao cho chi phí nối các dây lại với nhau là ít nhất.

Input:

- Số lượng dây: 4
- Độ dài dây L[]= { 4, 3, 2, 6}

Output: Chi phí nối dây nhỏ nhất.

OPT = 39

Chi phí nhỏ nhất được thực hiện như sau: lấy dây số 3 nối với dây số 2 để được tập 3 dây với độ dài 4, 5, 6. Lấy dây độ dài 4 nối với dây độ dài 5 ta nhận được tập 2 dây với độ dài 6, 9. Cuối cùng nối hai dây còn lại ta nhận được tập một dây với chi phí là 6+9 =15. Như vậy, tổng chi phí nhỏ nhất của ba lần nối dây là 5 + 9 + 15 = 29.

Ta không thể có cách nối dây khác với chi phí nhỏ hơn 39. Ví dụ lấy dây 1 nối dây 2 ta nhận được 3 dây với độ dài {7, 2, 6}. Lấy dây 3 nối dây 4 ta nhận được tập hai dây với độ dài {7, 8}, nối hai dây cuối cùng ta nhận được 1 dây với độ dài 15. Tuy vậy, tổng chi phí là 7 + 8 + 15 = 30.

Thuật toán. *Sử dụng phương pháp tham lam dựa vào hàng đợi ưu tiên*.

Thuật toán Greedy-N-Ropes(int L[], int n):

Input:

- n : số lượng dây.
- L[] : chi phí nối dây.

Output:

- Chi phí nối dây nhỏ nhất.

Actions:

Bước 1 . Tạo pq là hàng đợi ưu tiên lưu trữ độ dài n dây.

Bước 2 (Lặp).

```
OPT = 0; // Chi phí nhỏ nhất.
While (pq.size>1) {
    First = pq.top; pq.pop(); //Lấy và loại phần tử đầu tiên trong pq.
    Second = pq.top; pq.pop(); //Lấy và loại phần tử kế tiếp trong pq.
    OPT = First + Second; //Giá trị nhỏ nhất để nối hai dây
    Pq.push(First + Second); //Đưa lại giá trị First + Second vào pq.
}
```

Bước 3(Trả lại kết quả).

```
Return(OPT);
```

EndActions.

Ví dụ. Thuật toán Greedy-N-Ropes(int L[], int n):

Input:

- Số lượng dây n = 8.
- Chi phí nối dây L[] = { 9, 7, 12, 8, 6, 5, 14, 4}.

Output:

- Chi phí nối dây nhỏ nhất.

Phương pháp tiến hành::

Bước	Giá trị First, Second	OPT=?	Trạng thái hàng đợi ưu tiên.
		0	4, 5, 6, 7, 8, 9, 12, 14
1	First=4; Second=5	9	6, 7, 8, 9, 9, 12, 14
2	First=6; Second=7	22	8, 9, 9, 12, 13, 14
3	First=8; Second=9	39	9, 12, 13, 14, 17
4	First=9; Second=12	60	13, 14, 17, 21
5	First=13; Second=14	87	17, 21, 27
6	First=17; Second=21	125	27, 38
7	First=27; Second=38	190	65
OPT = 190			

2.5.5. Bài toán sắp đặt lại xâu ký tự

Bài toán. Cho xâu ký tự s[] độ dài n và số tự nhiên d. Hãy sắp đặt lại các ký tự trong xâu s[] sao cho hai ký tự giống nhau đều cách nhau một khoảng là d. Nếu bài toán có nhiều nghiệm, hãy đưa ra một cách sắp đặt đầu tiên tìm được. Nếu bài toán không có lời giải hãy đưa ra thông báo “Vô nghiệm”.

Ví dụ.

Input:

- Xâu ký tự S[] =“ABB”;
- Khoảng cách d = 2.

Output: BAB

Input:

- Xâu ký tự S[] =“AAA”;
- Khoảng cách d = 2.

Output: Vô nghiệm.

Input:

- Xâu ký tự S[] =“GEEKSFORGEEKS”;
- Khoảng cách d = 3.

Output: EGKEGKESFESFO

2.5.5. Bài toán sắp đặt lại xâu ký tự

Thuật toán Greedy-Arrang-String (S[], d):

Input:

- Xâu ký tự S[].
- Khoảng cách giữa các ký tự d.

Output: Xâu ký tự được sắp đặt lại thỏa mãn yêu cầu bài toán.

Formats : Greedy-Arrang-String(S, d, KQ);

Actions:

Bước 1. Tìm Freq[] là số lần xuất hiện mỗi ký tự trong xâu.

Bước 2. Sắp xếp theo thứ tự giảm dần theo số xuất hiện ký tự.

Bước 3. (Lặp).

```
i = 0; k =<Số lượng ký tự trong Freq[]>;
While ( (i < k ) {
    p = Max(Freq); //Chọn ký tự xuất hiện nhiều lần nhất.
    For ( t = 0; t<p; t++ ) // điền các ký tự i, i+d, i+2d, ..,i + pd
        if (i+(t*d)>n ) { < Không có lời giải>; return;>
        KQ[i + (t*d)]  = Freq[i].kytu;
    }
    i++;
}
```

Bước 4(Trả lại kết quả): Return(KQ);

EndActions.

Greedy-Arrang-String (S[], d):Input : S[] = “GEEKSFORGEEKS; d= 3.

Bước 1. Tìm tập ký tự và số lần xuất hiện mỗi ký tự.

Freq[]	
G	2
E	4
K	2
S	2
F	1
O	1
R	1

Sắp xếp theo thứ tự giảm dần của số lần xuất hiện.

Bước 2. Sắp xếp theo thứ tự giảm dần số lần xuất hiện.

Freq[]	
E	4
G	2
K	2
S	2
F	1
O	1
R	1

Bước 3. Lặp.

	KQ[I + p*d]												
I =0	E			E			E			E			
i=1	E	G		E	G		E			E			
i=2	E	G	K	E	G	K	E			E			
i=3	E	G	K	E	G	K	E	S		E	S		
i=4	E	G	K	E	G	K	E	S	F	E	S		
i=5	E	G	K	E	G	K	E	S	F	E	S	O	
i=6	E	G	K	E	G	K	E	S	F	E	S	O	R

BÀI 1. TÌM MAX

Cho mảng A[] gồm N phần tử.Nhiệm vụ của bạn là tìm $max = \sum_{i=0}^{n-1} A_i * i$ bằng cách sắp đặt lại các phần tử trong mảng. Chú ý, kết quả của bài toán có thể rất lớn vì vậy bạn hãy đưa ra kết quả lấy modulo với 10^9+7.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N; dòng tiếp theo đưa vào N số A[i] tương ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, A[i] \leq 10^7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 5 5 3 2 4 1 3 1 2 3	40 8

BÀI 2. TỔNG NHỎ NHẤT

Cho mảng A[] gồm các số từ 0 đến 9. Nhiệm vụ của bạn là tìm tổng nhỏ nhất của hai số được tạo bởi các số trong mảng A[]. Chú ý, tất cả các số trong mảng A[] đều được sử dụng để tạo nên hai số.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N; dòng tiếp theo đưa vào N số A[i] tương ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 20$; $0 \leq A[i] \leq 9$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 6 6 8 4 5 2 3 5 5 3 0 7 4	604 82

BÀI 3. CHIA MẢNG

Cho mảng A[] gồm N số nguyên không âm và số K. Nhiệm vụ của bạn là hãy chia mảng A[] thành hai mảng con có kích cỡ K và N-K sao cho hiệu giữa tổng hai mảng con là lớn nhất. Ví dụ với mảng A[] = {8, 4, 5, 2, 10}, K=2 ta có kết quả là 17 vì mảng A[] được chia thành hai mảng {4, 2} và { 8, 5,10} có hiệu của hai mảng con là 23-6=17 là lớn nhất.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N và số K; dòng tiếp theo đưa vào N số A[i] tương ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, K, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq K < N \leq 50$; $0 \leq A[i] \leq 1000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 5 2 8 4 5 2 10 8 3 1 1 1 1 1 1 1 1	17 2

BÀI 4. SẮP XẾP THAM LAM

Cho mảng A[] gồm N số và thực hiện các thao tác theo nguyên tắc dưới đây:

- Ta chọn một mảng con sao cho phần tử ở giữa của mảng con cũng là phần tử ở giữa của mảng A[] (trong trường hợp N lẻ).
- Đảo ngược mảng con đã chọn trong mảng A[]. Ta được phép chọn mảng con và phép đảo ngược mảng con bao nhiêu lần tùy ý.

Ví dụ với mảng A[] = {1, 6, **3, 4, 5**, 2, 7} ta có câu trả lời là Yes vì: ta chọn mảng con {3, 4, 5} và đảo ngược để nhận được mảng A[]={1, 6, **5, 4, 3**, 2, 7}, chọn tiếp mảng con {6, 5, 4, 3, 2} và đảo ngược ta nhận được mảng A[]={1, 2, 3, 4, 5, 6, 7}. Hãy cho biết ta có thể sắp xếp được mảng A[] bằng cách thực hiện các thao tác kể trên hay không?

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N; dòng tiếp theo đưa vào N số A[i] tương ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \le T \le 100$; $1 \le N \le 50$; $0 \le A[i] \le 1000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 7 1 6 3 4 5 2 7 7 1 6 3 4 5 7 2	Yes No

BÀI 5. GIÁ TRỊ NHỎ NHẤT CỦA BIỂU THỨC

Cho mảng A[], B[] đều có N phần tử. Nhiệm vụ của bạn là tìm giá trị nhỏ nhất của biểu thức P = A[0]*B[0] + A[1]*B[1] + ..+A[N-1]*B[N-1] bằng cách tráo đổi vị trí các phần tử của cả mảng A[] và B[].

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng thứ nhất đưa vào số phần tử của mảng N; dòng tiếp theo đưa vào N số A[i]; dòng cuối cùng đưa vào N số B[i] các số được viết cách nhau một vài khoảng trống.
- T, N, A[i], B[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^7$; $0 \leq A[i], B[i] \leq 10^{18}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 7 1 6 3 4 5 2 7 1 1 1 2 3 4 3 7 1 6 3 5 5 2 2	45 27

BÀI 6. SẮP XẾP CÔNG VIỆC

Cho hệ gồm N hành động. Mỗi hành động được biểu diễn như một bộ đôi <S_i, F_i> tương ứng với thời gian bắt đầu và thời gian kết thúc của mỗi hành động. Hãy tìm phương án thực hiện nhiều nhất các hành động được thực hiện bởi một máy hoặc một người sao cho hệ không xảy ra mâu thuẫn.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng thứ nhất đưa vào số lượng hành động N; dòng tiếp theo đưa vào N số S_i tương ứng với thời gian bắt đầu mỗi hành động; dòng cuối cùng đưa vào N số F_i tương ứng với thời gian kết thúc mỗi hành động; các số được viết cách nhau một vài khoảng trống.
- T, N, S_i, F_i thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, F_i, S_i \leq 1000$.

Output:

- Đưa số lượng lớn nhất các hành động có thể được thực thi bởi một máy hoặc một người.

Input	Output
1 6 1 3 0 5 8 5 2 4 6 7 9 9	4

BÀI 7. PHÂN SỐ ĐƠN VỊ

Một phân số đơn vị nếu tử số của phân số đó là 1. Mọi phân số nguyên dương đều có thể biểu diễn thành tổng các phân số đơn vị. Ví dụ 2/3 = 1/2 + 1/6. Cho phân số nguyên dương P/Q bất kỳ (P < Q), hãy biểu diễn phân số nguyên dương thành tổng phân số đơn vị.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là bộ đôi tử số P và mẫu số Q của phân số nguyên dương được viết trên một dòng.
- T, P, Q thỏa mãn ràng buộc: $1 \le T \le 100$; $1 \le P, Q \le 100$.

Output:

- Đưa ra đáp án tìm được trên 1 dòng, theo dạng "1/a + 1/b + …"

Input	Output
2 2 3 1 3	1/2 + 1/6 1/3

BÀI 8. XẾP LỊCH

Cho mảng A[] ghi lại N việc. Mỗi việc trong A[] được biểu diễn như một bộ 3 số nguyên dương <JobId, Deadline, Profit>, trong đó JobId là mã của việc, Deadline là thời gian kết thúc của việc, Profit là lợi nhuận đem lại nếu hoàn thành việc đó đúng thời gian. Thời gian để hoàn toàn mỗi công việc là 1 đơn vị thời gian. Hãy cho biết lợi nhuận lớn nhất có thể thực hiện các việc với giả thiết mỗi việc được thực hiện đơn lẻ.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là số lượng Job N; phần thứ hai đưa vào 3×N số tương ứng với N job.
- T, N, JobId, Deadline, Profit thỏa mãn ràng buộc:$1 \le T \le 100$; $1 \le N \le 1000$; $1 \le$ JobId ≤ 1000; $1 \le$ Deadline ≤ 1000; $1 \le$ Profit ≤ 1000.

Output:

- Đưa số lượng Job và lợi nhuận lớn nhất có thể đạt được.

Input	Output
1 4 1 4 20 2 1 10 3 1 40 4 1 30	2 60

BÀI 9. BIỂU THỨC ĐÚNG

Cho một mảng S gồm 2×N ký tự, trong đó có N ký tự '[' và N ký tự ']'. Xâu S được gọi là viết đúng nếu S có dạng S2[S1] trong đó S, S2 là các xâu viết đúng. Nhiệm vụ của bạn là tìm số các phép đổi chỗ ít nhất các ký tự kề nhau của xâu S viết sai để S trở thành viết đúng. Ví dụ với xâu S ="[]][][" ta có số phép đổi chỗ kề nhau ít nhất là 2.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu S viết sai theo nguyên tắc kể trên.
- T, S thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq length(S) \leq 100000$.

Output:

- Đưa số lượng Job và lợi nhuận lớn nhất có thể đạt được.

Input	Output
2 []][][[][][]	2 0

BÀI 10. NỐI DÂY

Cho N sợi dây với độ dài khác nhau được lưu trong mảng A[]. Nhiệm vụ của bạn là nối N sợi dây thành một sợi sao cho tổng chi phí nối dây là nhỏ nhất. Biết chi phí nối sợi dây thứ i và sợi dây thứ j là tổng độ dài hai sợi dây A[i] và A[j].

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào số lượng sợi dây N; dòng tiếp theo đưa vào N số A[i] là độ dài của các sợi dây; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \le T \le 100$; $1 \le N \le 10^6$; $0 \le A[i] \le 10^6$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 4 4 3 2 6 5 4 2 7 6 9	29 62