Đề cương ôn tập phân tích thiết kế thuật toán (CNTTVA1&CNTTVA2)

1. Hình thức thi viết 90 phút gồm 03 bài trong đề cương và 01 bài nâng cao
2. Yêu cầu, nêu được ý tưởng thuật toán, tính toán minh họa được kết quả của thuật toán bằng tay, viết được code
3. Nội dung gồm các bài sau

I. Thuật toán tham lam

* 1. Búp bê Nga ok
  2. Buôn dưa lê
  3. Giao hàng
  4. Làm bóng tuyết

II. Thuật toán quy hoạch động

* 1. Bài toán đổi tiền
  2. Xâu con chung dài nhất
  3. Sắp xếp ba lô
  4. Bảng số

III. Quay lui

* 1. Đổi tiền
  2. Dãy con liên tục có tổng lớn nhất
  3. Người đi du lịch

IV. Chia để trị

* 1. xâu Fibonacci ok
  2. Cắt thanh kim loại ok
  3. Đếm số nghịch thế (cây IT)

Bài làm

# I, Tham lam

## Búp bê nga.

### a, Ý tưởng

* Lồng các búp bê Nga vào nhau theo nguyên tắc con bé hơn sẽ nằm trong con to hơn. Sau đó tính xem có bao nhiêu con búp bê ngoài cùng và tổng kích thước.
* Phạm vi kết quả: Kết quả phải thuộc trong khoảng từ 1 đến 1e6.
* Sử dụng thuật toán tham lam:
  + VD: Có 8 con búp bê và kích thước tối đa chênh lệch để có thể lồng vào nhau là 2, biến res = 0.

n = 8, k = 2

4 7 2 8 4 8 3 2

* + Đưa 8 con búp bê vào hàng đợi ưu tiên.
  + Hàng đợi ưu tiên phải được sắp xếp theo thứ tự giảm dần
  + Kiểm tra xem phần tử đầu tiên tức 8 có lớn hơn hoặc bằng các phần tử khác trong queue + kích thước k không. Nếu lớn hơn thì pop phần tử được xét đó ra, ngược lại sẽ được ghép vs nhau.
  + Thực hiện vòng lặp như vậy cho đến khi kết thúc bài toán.
  + Kết quả sẽ là số phần tử còn sót lại trong queue và tổng kích thước của nó.

### b, Giải tay

* 8 8 7 4 4 3 2 2
* 8 >= 8+2(S) => res += 8
* 8 >= 7+2(Đ) => res += 8
* 8>= 7+2(S) => res += 7
* 8 >= 4+2(Đ) => pop(8)
* …
* 8 8 7 (size của queue)
* Số búp bê ngoài cùng = 3, tông kích thước = 23.

### c, Code

#include<bits/std++.h>

Using namespace std;

Int main(){

int n, k, res = 0;

cin>>n>>k;

int a[n+5];

priority\_queue<int> q;

for(int i = 1; i<=n; i++){

cin>>a[i];

}

sort(a+1, a+n+1, greater<int>());

for(int i = 1; i<=n; i++){

q.push(a[i]);

if(q.top() >= a[i] + k){

q.pop();

}else{

res += a[i];

}

}

cout<<q.size()<<” “<<res<<endl;

}

## Buôn dưa lê

### a, Ý tưởng:

* Tính tổng sản lượng lớn nhất của Tito thu hoạch trên cánh đồng biết năng lực của Tito có hạn và không thể một lúc thu hoạch hết được.
* Sử dụng thuật toán tham lam:
  + VD: Có 6 thửa ruộng, số ngày tối đa mà dưa phải thu hoạch là 2 và năng lực của Tito là 5. Đồng thời dòng thứ 2 là dữ liệu về sản lượng lần lượt dưa lê chín trong từng ngày 1 đến n.

n = 6, k = 2, m = 5

4 7 2 18 1 10.

* + Dùng hàng đợi.
  + Push các dữ liệu về sản lượng lần lượt dưa lê chín trong từng ngày từ 1 đến n vào queue.
  + Thực hiện vòng lặp kiểm tra xem chừng nào size trong queue mà lớn hơn k, tức là số ngày quá dài thì sẽ pop ra khỏi queue.
  + Tiếp tục kiểm tra trong vòng lặp xem chừng nào thu hết các thửa trong khả năng của mình => Tính tổng sản lượng, đồng thời pop ra khỏi queue.
  + Cuối cùng kiểm tra xem còn thừa thửa ruộng nào không, nếu còn thì sẽ tiếp tục thu hoạch dựa vô khả năng của mình.

### b, Giải tay

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | 7 | 2 | 18 | 1 | 10 |

Ngày 1: Thu được 5 mà dưa lê chin 4

=>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 7 | 2 | 18 | 1 | 10 |

4 7 2 18 1 10

Ngày 2: Thu được 5 mà dưa lê chin 7

=>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 2 | 2 | 18 | 1 | 10 |

4 5 2 18 1 10

Ngày 3: Thu được 5 mà dưa lê chin 2 ở hnay và 2 ở hôm qua

=>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 18 | 1 | 10 |

4 5 4 18 1 10

Ngày 4: Thu được 5 mà dưa lê chin 18

=>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 13 | 1 | 10 |

4 5 4 5 1 10

Ngày 5: Thu được 5 mà dưa lê chin 1 => Thu hoạch tiếp ngày 4 vì để quá 2 ngày sẽ bị hỏng => Thu hoạch 5 sp ở ngày 4 và số còn lại bị hỏng

=>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 8 | 1 | 10 |

4 5 4 5 5 10

Ngày 6: Thu được 5 mà dưa lê chin 10 hnay => Thu hoạch 1 sp từ ngày 5 và 4 sp hnay

=>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 8 | 0 | 6 |

4 5 4 5 5 5

Ngày 7: Thu được 5 mà dưa lê chin 6 ở ngày => Thu hoạch được 5 và 1 quả bị hỏng

=>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 8 | 0 | 1 |

4 5 4 5 5 5 5

=> KQ: Tổng sản lượng: 4+5+4+5+5+5+5 = 33

### c, Code

#include<bits/stdc++.h>

using namespace std;

int main(){

//n: So thua ruong, k: so ngay toi da, m: san luong

long long n, k, m, a[200005]={}, res = 0, t=0;

queue<long long> q;

cin>>n>>k>>m;

for(int i = 1; i<=n; i++){

cin>>a[i];

}

for(int i = 1; i<n+k; i++){

q.push(a[i]);

while(q.size()>k){// so ngay qua dai=> bo di

q.pop();

}

t=0;

while(q.size() && t+q.front()<=m){// thu het cac thua

t+=q.front();

q.pop();

}

if(q.size()){// neu van con thua chin, thu cho du m

q.front() -= m-t;

t=m;

}

res += t;

}

cout<<res;

}

## Giao hàng

### a, Ý tưởng

* Có n món hàng phải giao và mỗi đơn hàng có thời gian tối đa phải giao đến và mức thưởng. Tính mức thưởng cao nhất có thể của Titi.
* n: Số lượng đơn hàng, x: Thời gian tối đa, y: Mức thưởng
* Sử dụng thuật toán tham lam:
  + Sử dụng vector và hàng đợi ưu tiên.
  + Pair mỗi đơn hàng với mỗi thời gian tối đa và mức thưởng với nhau.
  + Chạy vòng lặp với thời gian tối đa giảm dần.
  + Push các mốc thời gian tối đa của các sp vào queue theo đk ở trên.
  + Kiểm tra size của queue, nếu không null thì kết quả sẽ là tổng các phần tử có trong queue, đồng thời pop khỏi queue.

### b, Giải tay

6

3 5

3 7

1 3

2 4

2 2

4 1

* 4 1

3 7

3 5

2 4

2 2

1 3

Giờ thứ nhất chuyển món thứ tư có thời hạn trong 2 giờ nhưng giao luôn giờ thứ nhất bạn được thưởng **4** tiền

Giờ thứ hai và ba chuyển món thứ nhất và thứ hai có thời hạn trong 3 giờ bạn giao một món đúng giờ và một món sớm giờ tổng cộng **12** tiền thưởng

Giờ thứ tư tất nhiên bạn giao món thứ sáu thời hạn trong 4 giờ và bạn vẫn kịp bỏ túi **1** tiền thưởng. => 17 tiền thưởng

### c, Code

#include<bits/stdc++.h>

using namespace std;

int main(){

int n, x, y;

long long res = 0;

cin>>n;

vector<long long> A[10006];

priority\_queue<long long> Q;

for(int i = 1; i<=n; i++){

cin>>x>>y;

A[x].push\_back(y);

}

for(int i = 1e4; i>0; i--){

for(auto x: A[i]){

Q.push(x);

}

if(Q.size()){

res += Q.top();

Q.pop();

}

}

cout<<res;

}

## Làm bóng tuyết

### a, Ý tưởng

* Có n ngày, m quả bóng tuyết và nhiệt độ mỗi ngày khiến cho trọng lượng của quả cầu tuyết ở mỗi ngày đó giảm đúng bằng 1 lượng nhiệt độ của ngày đó.
* Kết quả là trọng lượng của các quả cầu tuyết sau n ngày bị tan chảy vs t độ.
* Sử dụng hàng đợi ưu tiên sắp xếp tăng dần.
* Push các khối lượng của cầu tuyết vô hàng đợi.
* Chạy vòng lặp kiểm tra xem queue có rỗng hay không và phần tử đầu tiên trong queue trừ đi nhiệt độ hiện tại có nhỏ hơn nhiệt độ các ngày khác không. Nếu có thì tính tổng khối lượng các quả cầu tuyết sau khi trừ đi nhiệt độ trong ngày. Đồng thời nếu như nhiệt độ trong ngày không đủ làm tan hết thì sẽ tiếp tục cộng vô cho ngày tiếp theo

### b, Giải tay

n = 3

v[i] = 10 10 5

t[i] = 5 7 2

* Ngày 1: 5 10 5

0 7 2

* Ngày 2: 5 12 5

1. 2 2

* Ngày 3: 5 12 4

0 0 0

### c, Code

#include<bits/stdc++.h>

using namespace std;

struct cmp{

bool operator()(long long a, long long b){

return a>b;

}

};

int main(){

long long n, v[100005], t[100005];

long long p = 0;

priority\_queue<long long, vector<long long>, cmp> Q;

cin>>n;

for(int i = 1; i<=n; i++){

cin>>v[i];

}

for(int i = 1; i<=n; i++){

cin>>t[i];

}

for(int i = 1; i<=n; i++){

long long s = 0;

Q.push(p+v[i]);

while(Q.size() && Q.top() - p <= t[i]){

s += Q.top() - p;

Q.pop();

}

s += Q.size() \* t[i];

cout<<s<<" ";

p+=t[i];

if(Q.size()==0){

p = 0;

}

}

}