GREEDY!

Dont worry its definitely not cancer

What is greedy?

Basically just take best at current position lol

> *“A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. In many problems, a greedy strategy does not produce an optimal solution, but a greedy heuristic can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.”*

Greedy - proving it

Essentially we want to prove that the **local optimum guarantees the overall optimum.**

How will we do this? ~~Math~~

- Stay ahead argument
- Bounding argument
- Exchange argument

Stay ahead argument

This argument wants to argue that at every step of the greedy algorithm, it is **at least as good** as the optimum solution.

Stay ahead argument

This argument wants to argue that at every step of the greedy algorithm, it is **at least as good** as the optimum solution.

Generally :

- Find a series of measurements M_1, M_2, …, M□ you can apply to any solution

Stay ahead argument

This argument wants to argue that at every step of the greedy algorithm, it is **at least as good** as the optimum solution.

Generally :

- Find a series of measurements M_1, M_2, …, M□ you can apply to any solution
- Show that the greedy algorithm's measures are at least as good as any optimal solution's measures (This usually involves induction)

Stay ahead argument

This argument wants to argue that at every step of the greedy algorithm, it is **at least as good** as the optimum solution.

Generally :

- Find a series of measurements M_1, M_2, …, M□ you can apply to any solution
- Show that the greedy algorithm's measures are at least as good as any optimal solution's measures (This usually involves induction)
- Prove that because the greedy solution's measures are at least as good as any solution's measures, the greedy solution must be optimal (This is usually a proof by contradiction)

Application - lunchbox

Go do lunchbox!

Hahaha get skemmed again

Just prove by ac lmao

Greedy - proving it

Essentially we want to prove that the **local optimum guarantees the overall optimum.**

How will we do this? ~~Math~~

- ~~Stay ahead argument~~
- ~~Bounding argument~~
- ~~Exchange argument~~
- **Proof by AC**

Abridged problem statement

Given n lunchboxes, find the maximum number of schools you can give lunchboxes to.

Abridged problem statement

Given n lunchboxes, find the maximum number of schools you can give lunchboxes to.

Or this…

Shi tianle is giving C people conduct slips. He can give each person C_i L[i] or 0 conduct slips. Since he is not a teacher and is not supposed to be given conduct slips, he will get caught by the teacher if he gives more than K conduct slips and be given a conduct slip! (OH NOOO) Tianle does not want to be given a conduct slip, so help tianle maximise the number of people he can give conduct slips to!

Abridged problem statement

Given n lunchboxes, find the maximum number of schools you can give lunchboxes to.

Let i_1, i_2, …, i_{grd} be the indices of schools the greedy algorithm picks. Assume $K[i_1] \leq K[i_2] \leq \ldots \leq K[i_{grd}]$.
Let j_1, j_2, …, j_{opt} be the indices of schools that an arbitrary optimal algorithm picks. Assume $K[j_1] \leq K[j_2] \leq \ldots \leq K[j_{opt}]$.

Abridged problem statement

Given n lunchboxes, find the maximum number of schools you can give lunchboxes to.

Let i_1, i_2, …, i_{grd} be the indices of schools the greedy algorithm picks. Assume $K[i_1] \leq K[i_2] \leq \ldots \leq K[i_{grd}]$.

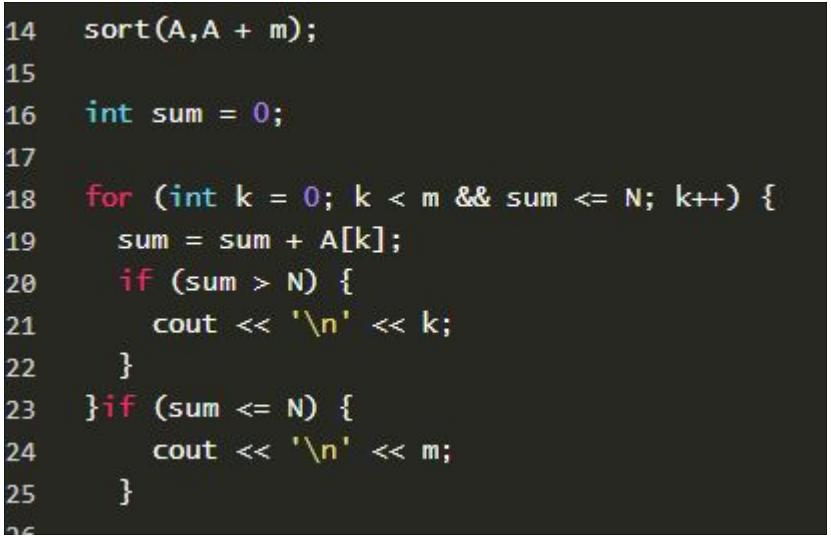
Let j_1, j_2, …, j_{opt} be the indices of schools that an arbitrary optimal algorithm picks. Assume $K[j_1] \leq K[j_2] \leq \ldots \leq K[j_{opt}]$.

$K[i_1] + K[i_2] + \ldots + K[i_{grd}] \leq N$

$K[j_1] + K[j_2] + \ldots + K[j_{opt}] \leq N$

Since we know that $K[i_x] \leq K[j_x]$ for every x, grd ≥ opt. Therefore the greedy algorithm is optimal.

Lunchbox - sample code



```
sort(A,A + m);

int sum = 0;

for (int k = 0; k < m && sum <= N; k++) {
  sum = sum + A[k];
  if (sum > N) {
    cout << '\n' << k;
  }
}if (sum <= N) {
    cout << '\n' << m;
  }
```

More greedy problems

Easy problems: potato salad, gss, paint

~~Harder~~ also easy problems: catlunch, bestplace, competition

Harder problems: Luarulers

Imple cancer problem: Lualectures

Luarulers - abridged problem statement

In an array a of n elements, maximise the number of a_i such that a[1]+a[2]+a[3]…+a_i[i]= 0. We are allowed to change the values of a[b[j]] where b is an array of length m for all j.

Luarulers - st2

m=1, b[1]=0.

We are allowed to change the value of the first element in the array a.

Luarulers - st2

m=1, b[1]=0.

We are allowed to change the value of the first element in the array a.

We create a prefix sums array for all a[i] and find the most number of repeated sums.

Luarulers - st3

m=1, b[1] can be anywhere in a.

We realise that changing the value a[b[1]] only affects values **after** a[b[1]].

Luarulers - st3

m=1, b[1] can be anywhere in a.

We realise that changing the value a[b[1]] only affects values **after** a[b[1]].

So we maintain the same prefix sum array and just count the most number of repeated sums starting from a[b[1]].

Remember to add those sums that are already 0 in the a[1] - a[b[1]] range.

Luarulers - st4

M≥1.

We observe that for each a[b[i]] only affects the range a[b[i]] to a[b[i+1]].

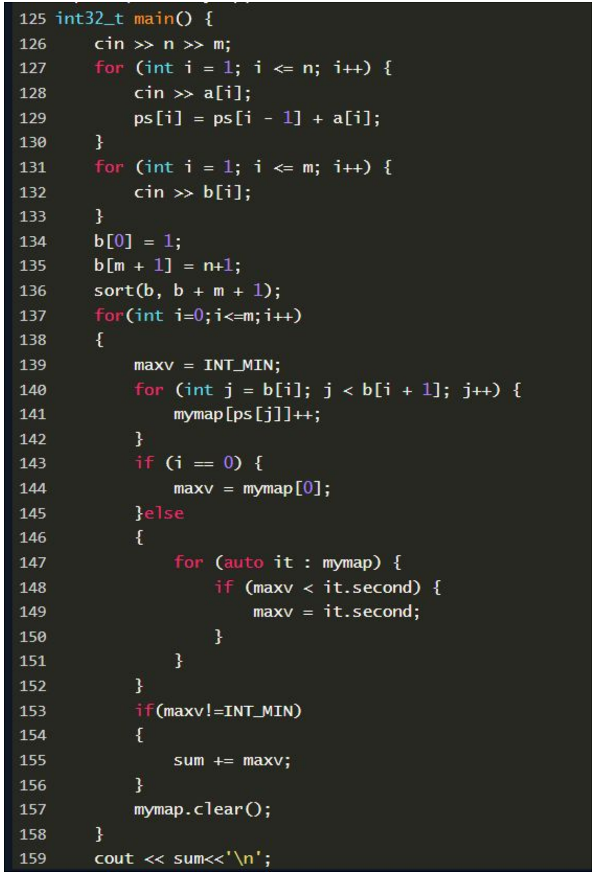
Luarulers - st4

M≥1.

We observe that for each a[b[i]] only affects the range a[b[i]] to a[b[i+1]]. We just compute the most repeated sum in the prefix sum array for all ranges a[b[i]] to a[b[i+1]].

Remember to add the number of 0s in the range 1 to a[b[i]].

Sample code



```
int32_t main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        ps[i] = ps[i - 1] + a[i];
    }
    for (int i = 1; i <= m; i++) {
        cin >> b[i];
    }
    b[0] = 1;
    b[m + 1] = n+1;
    sort(b, b + m + 1);
    for(int i=0;i<=m;i++)
    {
        maxv = INT_MIN;
        for (int j = b[i]; j < b[i + 1]; j++) {
            mymap[ps[j]]++;
        }
        if (i == 0) {
            maxv = mymap[0];
        }else
        {
            for (auto it : mymap) {
                if (maxv < it.second) {
                    maxv = it.second;
                }
            }
        }
        if(maxv!=INT_MIN)
        {
            sum += maxv;
        }
        mymap.clear();
    }
    cout << sum<<'\n';
```

Sample code

Haha no stop copying my code >:(

Ac or no lunch >:(

Easy?

The previous problems were quite easy right?

Greedy easy?

Cancer problem :O free snack if solve

Lualectures

Lualectures - Abridged problem statement

Find the minimum number of lecture halls needed to hold n lectures and output the designation of each lecture hall respectively.

Lualectures - Abridged problem statement

Find the minimum number of lecture halls needed to hold n lectures and output the designation of each lecture hall respectively.

Remember to give the assignment in order that the lectures are given by the question

Lualectures - solution

We observe that we can reuse a lecture hall if the start time of the lecture we are processing is more than the end time of a lecture that we have already processed.

Lualectures - solution

We observe that we can reuse a lecture hall if the start time of the lecture we are processing is more than the end time of a lecture that we have already processed. So, we will want to sort the lectures we have processed by their end time and store the lecture hall they used.

Lualectures - solution

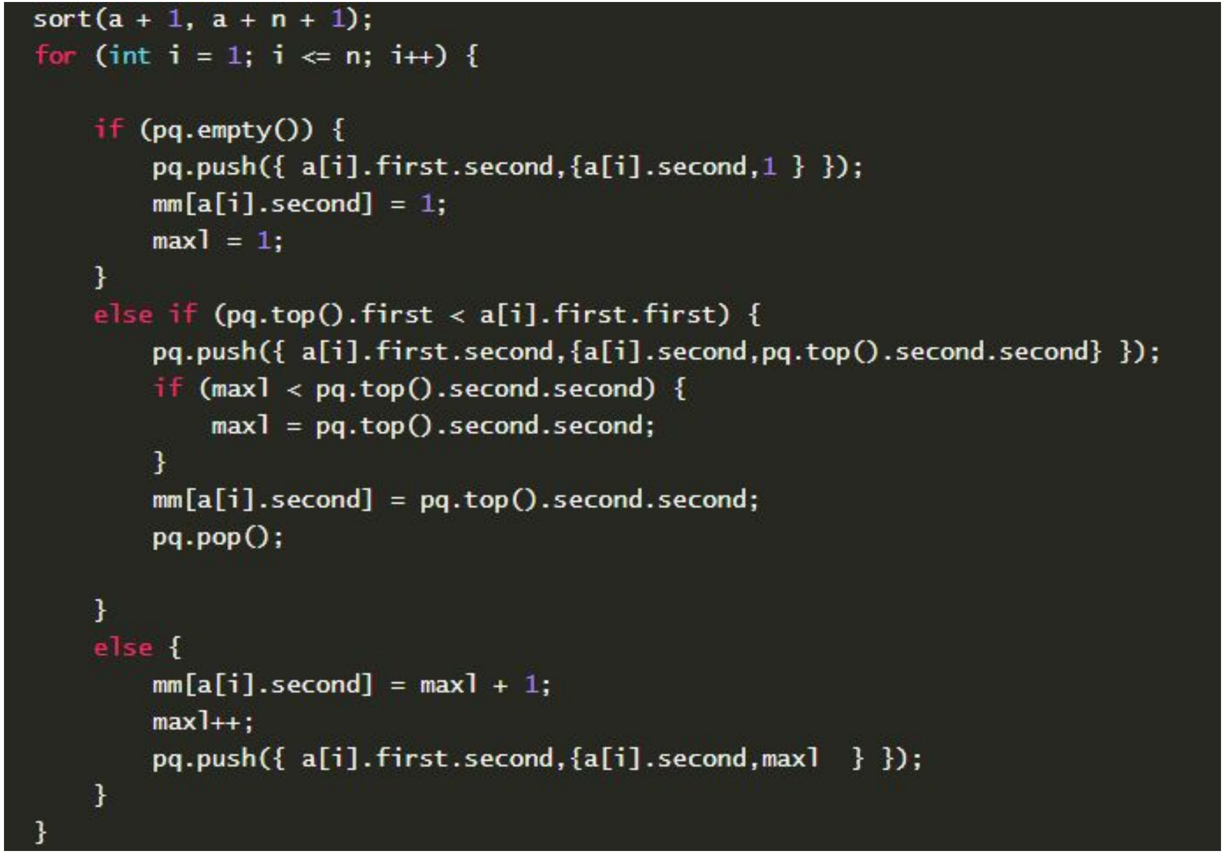
We observe that we can reuse a lecture hall if the start time of the lecture we are processing is more than the end time of a lecture that we have already processed. So, we will want to sort the lectures we have processed by their end time and store the lecture hall they used.

We then process lectures by their start time and keep track of ongoing lectures in a priority queue. We check that a lecture has ended when we process each new lecture and try to reuse halls as much as possible, otherwise we allocate a new lecture hall to the current lecture. We need to store this in a map as we need to give our answer in the order the question gives.

Lualectures - solution

In the end, iterate the map.

Sample code



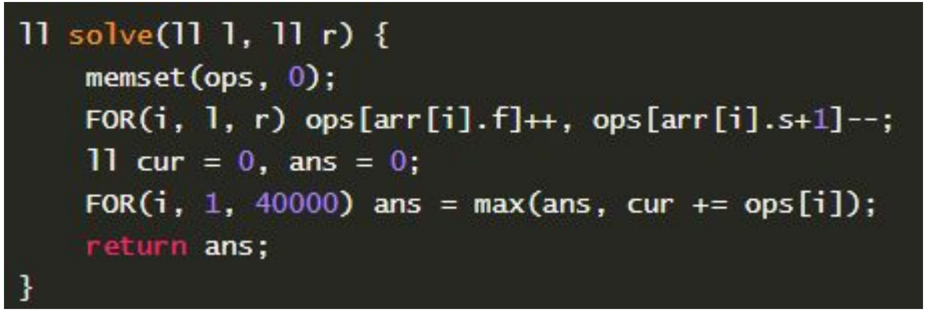
```
sort(a + 1, a + n + 1);
for (int i = 1; i <= n; i++) {

    if (pq.empty()) {
        pq.push({ a[i].first.second,{a[i].second,1 } });
        mm[a[i].second] = 1;
        maxl = 1;
    }
    else if (pq.top().first < a[i].first.first) {
        pq.push({ a[i].first.second,{a[i].second,pq.top().second.second} });
        if (maxl < pq.top().second.second) {
            maxl = pq.top().second.second;
        }
        mm[a[i].second] = pq.top().second.second;
        pq.pop();

    }
    else {
        mm[a[i].second] = maxl + 1;
        maxl++;
        pq.push({ a[i].first.second,{a[i].second,maxl  } });
    }
}
```

Fun!

Getting the maximum number of lectures is easy

Just range add 1 to [l, r] for all lecture halls and…

Range max (1, 1e9)



```
ll solve(ll l, ll r) {
    memset(ops, 0);
    FOR(i, l, r) ops[arr[i].f]++, ops[arr[i].s+1]--;
    ll cur = 0, ans = 0;
    FOR(i, 1, 40000) ans = max(ans, cur += ops[i]);
    return ans;
}
```

See workload2 (99 points).

Constructing tho… die.