Greedy Algorithm

Construct a solution iteratively, via sequence of myopic decisions, and hope that everything works out in the end.

Prepared by Pawan K. Mishra

Template

Greedy Algorithm(A,n)

```
    Candidates = rank (A)
    solution = ø
    for i = 1 to n
        c= findbest(Candidates)
        solution= solution U {c}
        candidates= candidates \ {c}
        candidates= revaluate(candidates)
    return solution
```

Sorting / or do something to rank
Initialize solution
Some greedy choice
Check feasibility before adding
Remove the selected element
Update the set candidates (optional)

Prepared by Pawan K. Mishra

Features and Bugs of the Greedy paradigm

- Easy to come up with one or more greedy algorithms
- Easy to analyse the running time
- Hard to establish correctness
- Warning: Most greedy algorithms are not always correct.

Exchange trick, to prove correctness (Worked often, but not always)

Let A be the greedy algorithm that we are trying to prove correct, and

A(I) the output of A on some input I.

Let O be an optimal solution on input I that is not equal to A(I).

The goal in exchange argument is to show how to modify O to create a new solution O’ with the following properties:

- 1. O’ is at least as good of solution as O (or equivalently O’ is also optimal), and
- 2. O’ is “more like” A(I) than O.

Exchange trick, to prove correctness (Worked often, but not always)

Let A be the greedy algorithm that we are trying to prove correct, and

A(I) the output of A on some input I.

Let O be an optimal solution on input I that is not equal to A(I).

The goal in exchange argument is to show how to modify O to create a new solution O’ with the following properties:

- 1. O’ is at least as good of solution as O (or equivalently O’ is also optimal), and
- 2. O’ is “more like” A(I) than O.

THIS IS THE CREATIVE PART - different for each algorithm/problem.

Two ways to proceed

First way. Contradiction

Theorem: The algorithm A solves the problem.

Proof by contradiction: Algorithm A doesn't solve the problem.

- Hence, there must be some input I on which A does not produce an optimal solution. Let the output produced by A be A(I).

Fact: Let O be the optimal solution that is most like A(I).

- If we can show how to modify O to create a new solution O' with the following properties:

1. O' is at least as good of solution as O (and hence O' is also optimal), and
2. O' is more like A(I) than O.

Then we have a contradiction to the choice of O. Thus, the theorem.

Two ways to proceed

Second way. Constructive way

Theorem: The algorithm A solves the problem.

Let I be an arbitrary instance. Let O be arbitrary optimal solution for I. Assume that we can show how to modify O to create a new solution O' with the following properties

1. O' is at least as good of solution as O (and hence O' is also optimal), and
2. O' is more like A(I) than O.

Then consider the sequence O'; O''; O'''; O''''; ....

Each element of this sequence is optimal, and more like A(I) than the proceeding element. Hence, ultimately this sequence must terminate with A(I).

Hence, A(I) is optimal.

P1: Job Scheduling

Time in the system for a job: Waiting time

All jobs arrived at time ZERO

IP: Set of n jobs=$\{j_1, j_2, \ldots j_n\}$ with processing time $P(j_1)$, $P(j_2)$, $\ldots P(j_n)$, and a single resource.

OP: Schedule jobs on one resource s.t. it minimizes the total waiting time in the system.

Example

Job 1- 5 units , Job 2- 10 units, Job 3- 4 units

Order waiting time

[1,2,3]- 0+(5)+(5+10)= 20

[3,1,2]- 0+(4)+(4+5)=13

Example

Job 1- 5 units , Job 2- 10 units, Job 3- 4 units

Order waiting time

[1,2,3]- 0+(5)+(5+10)= 20

[3,1,2]- 0+(4)+(4+5)=13… this order is the optimal

Developing Intuition

Some arbitrary order of jobs…

Finish time of job 1 = t_1

Finish time of job 2 = $t_1 + t_2$

Finish time of job 3 = $t_1 + t_2 + t_3$

.

.

.

Finish time of job n = $t_1 + t_2 + \ldots + t_n$

Total Finishing Time = $nt_1 + (n-1)t_2 + \ldots + t_n$

Developing Intuition

Some arbitrary order of jobs…

Finish time of job 1 = t_1

Finish time of job 2 = $t_1 + t_2$

Finish time of job 3 = $t_1 + t_2 + t_3$

Can you guess the greedy choice ?

.

.

.

Finish time of job n = $t_1 + t_2 + \ldots + t_n$

Total Finishing Time = $nt_1 + (n-1)t_2 + \ldots + t_n$

Developing Intuition

Some arbitrary order of jobs…

Finish time of job 1 = t_1

Finish time of job 2 = $t_1 + t_2$

Finish time of job 3 = $t_1 + t_2 + t_3$

.

.

.

Finish time of job n = $t_1 + t_2 + \ldots + t_n$

Total Finishing Time = $nt_1 + (n-1)t_2 + \ldots + t_n$

Can you guess the greedy choice ?

Shortest Job First

The only schedule that minimizes the total waiting time in the system is one that schedules jobs in nondecreasing order by processing time.

Use exchange trick……with contradiction

Assume the statement is not true.

Fact: OPT is an optimal solution.

Note that in OPT, there exists two consecutive jobs X and Y, s.t. X is being served before Y, and P(x) > P(y).

Else OPT will be same schedule which follows the shortest job sequence order.

Now suppose we interchange X and Y in OPT, and else remain same. What happens?

The only schedule that minimizes the total waiting time in the system is one that schedules jobs in nondecreasing order by processing time.

Use exchange trick…..with contradiction

Assume the statement is not true.

Fact: OPT is an optimal solution.

Note that in OPT, there exists two consecutive jobs X and Y, s.t. X is being served before Y, and P(x) > P(y).

Else OPT will be same schedule which follows the shortest job sequence order.

Now suppose we interchange X and Y in OPT, and else remain same. What happens?

Order in OPT   …………………. XY………………….

New order OPT' …………………. YX………………….

The only schedule that minimizes the total waiting time in the system is one that schedules jobs in nondecreasing order by processing time.

Use exchange trick…..with contradiction

Assume that it is not true.

Fact: OPT is an optimal solution.

Note that in OPT, there exists two consecutive jobs X and Y, s.t. X is being served before Y, and P(x) > P(y).

Else OPT will be same schedule which follows the shortest job sequence order.

Now suppose we interchange X and Y in OPT, and else remain same. What happens?

Order in OPT	………………… XY	…………………..	No need to worry about these jobs (think why??)
New order OPT’	………………… YX	…………………..	

The only schedule that minimizes the total waiting time in the system is one that schedules jobs in nondecreasing order by processing time..

Use exchange trick…..with contradiction

Assume that it is not true.

Fact: OPT is an optimal solution.

Note that in OPT, there exists two consecutive jobs X and Y, s.t. X is being served before Y, and P(x) > P(y).

Else OPT will be same schedule which follows the shortest job sequence order.

Now suppose we interchange X and Y in OPT, and else remain same. What happens?

Order in OPT	………………… XY……………………	In OPT, to do Job Y, need to wait till Job X being served.
New order OPT’	………………… YX……………………	In OPT’, to do Job X, need to wait till Job Y being served.

Prepared by Pawan K. Mishra

The only schedule that minimizes the total waiting time in the system is one that schedules jobs in nondecreasing order by processing time.

Use exchange trick…..with contradiction

Assume that it is not true.

Fact: OPT is an optimal solution.

Note that in OPT, there exists two consecutive jobs X and Y, s.t. X is being served before Y, and P(x) > P(y).

Else OPT will be same schedule which follows the shortest job sequence order.

Now suppose we interchange X and Y in OPT, and else remain same. What happens?

Order in OPT	…………………. XY…………………..	In OPT, to do Job Y, need to wait till Job X being served.
New order OPT'	…………………. YX…………………..	In OPT', to do Job X, need to wait till Job Y being served.

Total wait Time (OPT')= Total wait Time(OPT)-P(X)+P(Y)

The only schedule that minimizes the total waiting time in the system is one that schedules jobs in nondecreasing order by processing time.

Use exchange trick…..with contradiction

Assume that statement is not true.

Fact: OPT is an optimal solution.

Note that in OPT, there exists two consecutive jobs X and Y, s.t. X is being served before Y, and P(x) > P(y).

Else OPT will be same schedule which follows the shortest job sequence order.

Now suppose we interchange X and Y in OPT, and else remain same. What happens?

Order in OPT	………………… XY…………………	In OPT, to do Job Y, need to wait till Job X being served.
New order OPT’	………………… YX…………………	In OPT’, to do Job X, need to wait till Job Y being served.

Total wait Time (OPT’)= Total wait Time(OPT)-P(X)+P(Y) . Now, we know that P(x) > P(y).

Thus, Total wait Time (OPT’) < Total wait Time(OPT)

CONTADICTION TO THE OPTIMALITY

Job Scheduling with m resources, (minimize waiting time)

GENERALIZE THE PREVIOUS IDEA TO SOLVE THIS PROBLEM.

Flow to solve a problem

Given a problem P,

Try to come up with a greedy algorithm

a) then try to construct a counter example,

b) if you can construct, GOTO step 1.

c) if you cannot construct a counter example, ask your **friend.**

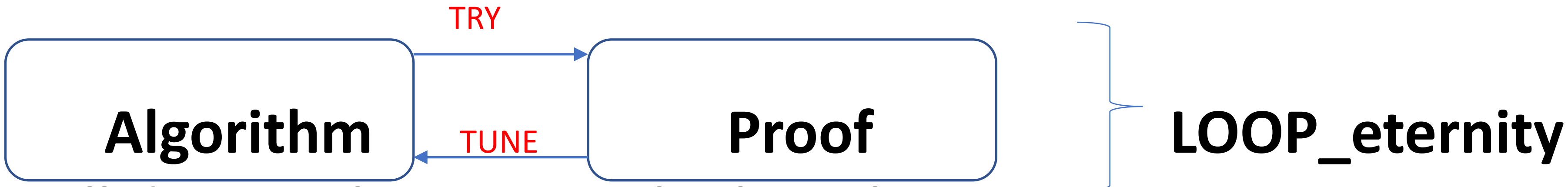
d) if your friend also fail to find one, ask **me**.

e) If we cannot construct one, we will post it in **math.stackexchange** **or** **mathoverflow.net,** and wait for few days..

f) if no one replies, then try to prove that your greedy algorithm works.

g) if we can prove, then alright.

h) if we are unable to prove, time to tune our greedy choice based on the difficulty we face during the proof.



i) I will discuss this step in the last class.