Greedy Algorithms: Minimizing Lateness

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 7

Greedy Algorithms

- Make a sequence of local choices to achieve a global optimum
- Never go back and revise an earlier decision
- How to prove that local choices achieve global optimum?

Greedy Algorithms

- Make a sequence of local choices to achieve a global optimum
- Never go back and revise an earlier decision
- How to prove that local choices achieve global optimum?

Strategy 1

- Greedy solution and optimal may not be identical — interval scheduling
- Incrementally show that the greedy solution is at least as good as an optimal one
- The greedy algorithm "stays ahead" of the optimal

Greedy Algorithms

- Make a sequence of local choices to achieve a global optimum
- Never go back and revise an earlier decision
- How to prove that local choices achieve global optimum?

Strategy 1

- Greedy solution and optimal may not be identical — interval scheduling
- Incrementally show that the greedy solution is at least as good as an optimal one
- The greedy algorithm "stays ahead" of the optimal

Strategy 2

- Greedy solution and optimal have a common structure
- Transform the optimal solution to match the greedy one, preserving optimality

Minimizing lateness

- IIT Madras has a single 3D printer

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer
- User i's item takes time $T(i)$ to print

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer
- User i's item takes time $T(i)$ to print
- User i has a deadline $D(i)$

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer
- User i's item takes time $T(i)$ to print
- User i has a deadline $D(i)$
- Each user will get access to the printer, but may not finish before deadline

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer
- User i's item takes time $T(i)$ to print
- User i has a deadline $D(i)$
- Each user will get access to the printer, but may not finish before deadline
- User i starts at time $S(i)$ and finishes at time $F(i) = S(i) + T(i)$

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer
- User i's item takes time $T(i)$ to print
- User i has a deadline $D(i)$
- Each user will get access to the printer, but may not finish before deadline

- User i starts at time $S(i)$ and finishes at time $F(i) = S(i) + T(i)$
- If $F(i) > D(i)$, user i is late by $L(i) = F(i) - D(i)$

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer
- User i's item takes time $T(i)$ to print
- User i has a deadline $D(i)$
- Each user will get access to the printer, but may not finish before deadline
- User i starts at time $S(i)$ and finishes at time $F(i) = S(i) + T(i)$
- If $F(i) > D(i)$, user i is late by $L(i) = F(i) - D(i)$
- Maximum lateness: $\max_i L(i)$

Minimizing lateness

- IIT Madras has a single 3D printer
- A number of users need to use this printer
- User i's item takes time $T(i)$ to print
- User i has a deadline $D(i)$
- Each user will get access to the printer, but may not finish before deadline
- User i starts at time $S(i)$ and finishes at time $F(i) = S(i) + T(i)$
- If $F(i) > D(i)$, user i is late by $L(i) = F(i) - D(i)$
- Maximum lateness: $\max_i L(i)$
- Goal Minimize the maximum lateness

Greedy strategies

Strategy 1 Schedule requests in increasing order of length — $T(i)$

Greedy strategies

Strategy 1 Schedule requests in increasing order of length — $T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 100$
  - $T(2) = 10$, $D(2) = 10$

Greedy strategies

Strategy 1 Schedule requests in increasing order of length — $T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 100$
  - $T(2) = 10$, $D(2) = 10$

Strategy 2 Give priority to requests with smaller slack time — $D(i) - T(i)$

Greedy strategies

Strategy 1 Schedule requests in increasing order of length — $T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 100$
  - $T(2) = 10$, $D(2) = 10$

Strategy 2 Give priority to requests with smaller slack time — $D(i) - T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 2$
  - $T(2) = 10$, $D(2) = 10$

Greedy strategies

Strategy 1 Schedule requests in increasing order of length — $T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 100$
  - $T(2) = 10$, $D(2) = 10$

Strategy 2 Give priority to requests with smaller slack time — $D(i) - T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 2$
  - $T(2) = 10$, $D(2) = 10$

Strategy 3 Schedule requests in increasing order of deadlines — $D(i)$

Greedy strategies

Strategy 1 Schedule requests in increasing order of length — $T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 100$
  - $T(2) = 10$, $D(2) = 10$

Strategy 2 Give priority to requests with smaller slack time — $D(i) - T(i)$

Counterexample

- Two jobs
  - $T(1) = 1$, $D(1) = 2$
  - $T(2) = 10$, $D(2) = 10$

Strategy 3 Schedule requests in increasing order of deadlines — $D(i)$

- This works, but how do you prove it is correct?

Correctness

- Renumber the jobs so that they are sorted by deadline
  - $D(1) \leq D(2) \leq \cdots \leq D(n)$

Correctness

- Renumber the jobs so that they are sorted by deadline
  - $D(1) \leq D(2) \leq \cdots \leq D(n)$
- Final schedule is simply $1, 2, \ldots, n$
  - Job 1 starts at $S(1) = 0$, ends at $F(1) = T(1)$
  - Job 2 starts at $S(2) = F(1)$, ends at $F(2) = S(2) + T(2)$
  - ...
  - Job n starts at $S(n) = F(n-1)$, ends at $F(n) = S(n) + T(n)$

Correctness

- Renumber the jobs so that they are sorted by deadline
  - $D(1) \leq D(2) \leq \cdots \leq D(n)$
- Final schedule is simply $1, 2, \ldots, n$
  - Job 1 starts at $S(1) = 0$, ends at $F(1) = T(1)$
  - Job 2 starts at $S(2) = F(1)$, ends at $F(2) = S(2) + T(2)$
  - ...
  - Job n starts at $S(n) = F(n-1)$, ends at $F(n) = S(n) + T(n)$
- Our schedule has no gaps
  - 3D printer is continuously in use from $S(1) = 0$ to $F(n)$
  - No idle time

Correctness

- Renumber the jobs so that they are sorted by deadline
  - $D(1) \leq D(2) \leq \cdots \leq D(n)$
- Final schedule is simply $1, 2, \ldots, n$
  - Job 1 starts at $S(1) = 0$, ends at $F(1) = T(1)$
  - Job 2 starts at $S(2) = F(1)$, ends at $F(2) = S(2) + T(2)$
  - ...
  - Job n starts at $S(n) = F(n-1)$, ends at $F(n) = S(n) + T(n)$
- Our schedule has no gaps
  - 3D printer is continuously in use from $S(1) = 0$ to $F(n)$
  - No idle time

Claim

There is an optimal schedule with no idle time

Correctness

- Renumber the jobs so that they are sorted by deadline
  - $D(1) \leq D(2) \leq \cdots \leq D(n)$
- Final schedule is simply $1, 2, \ldots, n$
  - Job 1 starts at $S(1) = 0$, ends at $F(1) = T(1)$
  - Job 2 starts at $S(2) = F(1)$, ends at $F(2) = S(2) + T(2)$
  - $\ldots$
  - Job n starts at $S(n) = F(n-1)$, ends at $F(n) = S(n) + T(n)$
- Our schedule has no gaps
  - 3D printer is continuously in use from $S(1) = 0$ to $F(n)$
  - No idle time

Claim

There is an optimal schedule with no idle time

- Eliminate idle time by shifting jobs earlier
- Can only reduce lateness

Correctness

Exchange argument

- Let O be some other optimal schedule
- Transform O to be identical to greedy schedule $1, 2, \ldots, n$

Correctness

Exchange argument

- Let O be some other optimal schedule
- Transform O to be identical to greedy schedule $1, 2, \ldots, n$

Inversions

- O has an inversion if i appears before j but $D(j) < D(i)$
- Greedy schedule $1, 2, \ldots, n$ has no inversions, by construction

Correctness

Exchange argument

- Let O be some other optimal schedule
- Transform O to be identical to greedy schedule $1, 2, \ldots, n$

Inversions

- O has an inversion if i appears before j but $D(j) < D(i)$
- Greedy schedule $1, 2, \ldots, n$ has no inversions, by construction

Claim

Any two schedules with no inversions and no idle time have same maximum lateness

Correctness

Exchange argument

- Let O be some other optimal schedule
- Transform O to be identical to greedy schedule $1, 2, \ldots, n$

Inversions

- O has an inversion if i appears before j but $D(j) < D(i)$
- Greedy schedule $1, 2, \ldots, n$ has no inversions, by construction

Claim

Any two schedules with no inversions and no idle time have same maximum lateness

- No inversions, idle time — only difference can be the order of requests with same deadline
- Reordering jobs with same deadline produces same lateness

Correctness

Exchange argument

- Let O be some other optimal schedule
- Transform O to be identical to greedy schedule $1, 2, \ldots, n$

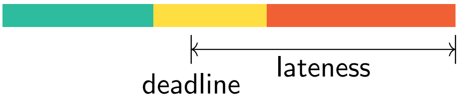
Inversions

- O has an inversion if i appears before j but $D(j) < D(i)$
- Greedy schedule $1, 2, \ldots, n$ has no inversions, by construction

Claim

Any two schedules with no inversions and no idle time have same maximum lateness

- No inversions, idle time — only difference can be the order of requests with same deadline
- Reordering jobs with same deadline produces same lateness



Correctness

Exchange argument

- Let O be some other optimal schedule
- Transform O to be identical to greedy schedule $1, 2, \ldots, n$

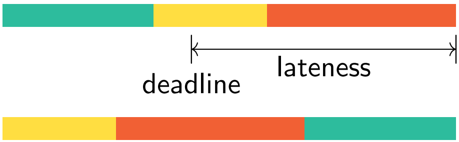
Inversions

- O has an inversion if i appears before j but $D(j) < D(i)$
- Greedy schedule $1, 2, \ldots, n$ has no inversions, by construction

Claim

Any two schedules with no inversions and no idle time have same maximum lateness

- No inversions, idle time — only difference can be the order of requests with same deadline
- Reordering jobs with same deadline produces same lateness



Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

A. If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
   - Find the first point where deadline decreases

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

A. If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
   - Find the first point where deadline decreases

B. Swap i and j to get one less inversion
   - Obvious

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

A. If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
   - Find the first point where deadline decreases

B. Swap i and j to get one less inversion
   - Obvious

C. Swapping i and j does not increase lateness of O
   - Not so obvious

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

A. If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
   - Find the first point where deadline decreases

B. Swap i and j to get one less inversion
   - Obvious

C. Swapping i and j does not increase lateness of O
   - Not so obvious

Correctness

Claim

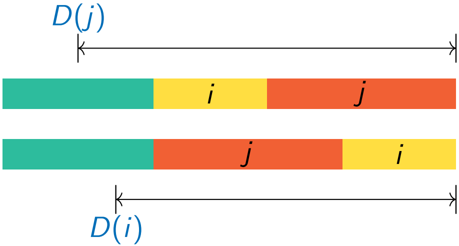
There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

A. If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
   - Find the first point where deadline decreases

B. Swap i and j to get one less inversion
   - Obvious

C. Swapping i and j does not increase lateness of O
   - Not so obvious
   - Recall that $D(j) < D(i)$



Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

- A If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
- B Swap i and j to get one less inversion
- C Swapping i and j does not increase lateness of O

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

- A If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
- B Swap i and j to get one less inversion
- C Swapping i and j does not increase lateness of O

- From $\boxed{C}$ we can remove each adjacent inversion without increasing lateness

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

A. If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$

B. Swap i and j to get one less inversion

C. Swapping i and j does not increase lateness of O

- From $\boxed{C}$ we can remove each adjacent inversion without increasing lateness
- At most $n(n-1)/2$ inversions in O to start with

Correctness

Claim

There is an optimal schedule with no inversions and no idle time

Let O be an optimal schedule with no idle time

- A. If O has an inversion, there are jobs i, j such that j is scheduled immediately after i and $D(j) < D(i)$
- B. Swap i and j to get one less inversion
- C. Swapping i and j does not increase lateness of O

- From $\boxed{C}$ we can remove each adjacent inversion without increasing lateness
- At most $n(n-1)/2$ inversions in O to start with
- Repeatedly remove adjacent inversions to get an optimal schedule with no inversions, no idle time

Summary

- Schedule requests with start times $T(i)$, deadlines $D(i)$, to minimize maximum lateness

Summary

- Schedule requests with start times $T(i)$, deadlines $D(i)$, to minimize maximum lateness
- Simple greedy algorithm with complexity $O(n \log n)$
  - Sort the requests by $D(i)$ — $O(n \log n)$
  - Read off schedule in sorted order — $O(n)$

Summary

- Schedule requests with start times $T(i)$, deadlines $D(i)$, to minimize maximum lateness
- Simple greedy algorithm with complexity $O(n \log n)$
  - Sort the requests by $D(i)$ — $O(n \log n)$
  - Read off schedule in sorted order — $O(n)$
- Correctness follows from an "exchange argument"
  - Consider any optimal solution O
  - Transform it, step by step, to be equal to the greedy solution