Lecture 33

CSE 331

Nov 17, 2017

Homework 9

Homework 9

Due by **11:00am, Friday, December 1, 2017**.

Make sure you follow all the homework policies.

All submissions should be done via Autolab.

Question 1 (Programming Assignment) [40 points]

</> Note

This assignment can be solved in either Java, Python or C++ (you should pick the language you are most comfortable with). Please make sure to look at the supporting documentation and files for the language of your choosing.

The Problem

In this problem, we will find closest pair of points on 2D plane.

! Note on Timeouts

For this problem the total timeout for Autolab is 480s, which is higher the the usual timeout of 180s in the earlier homeworks. So if your code takes a long time to run it'll take longer for you to get feedback on Autolab. **Please start early to avoid getting deadlocked out before the feedback deadline.**

Also for this problem, `C++` and `Java` are way faster. The 480s timeout was chosen to accommodate the fact that Python is much slower than these two languages.

Thanksgiving break

HW 8 solutions

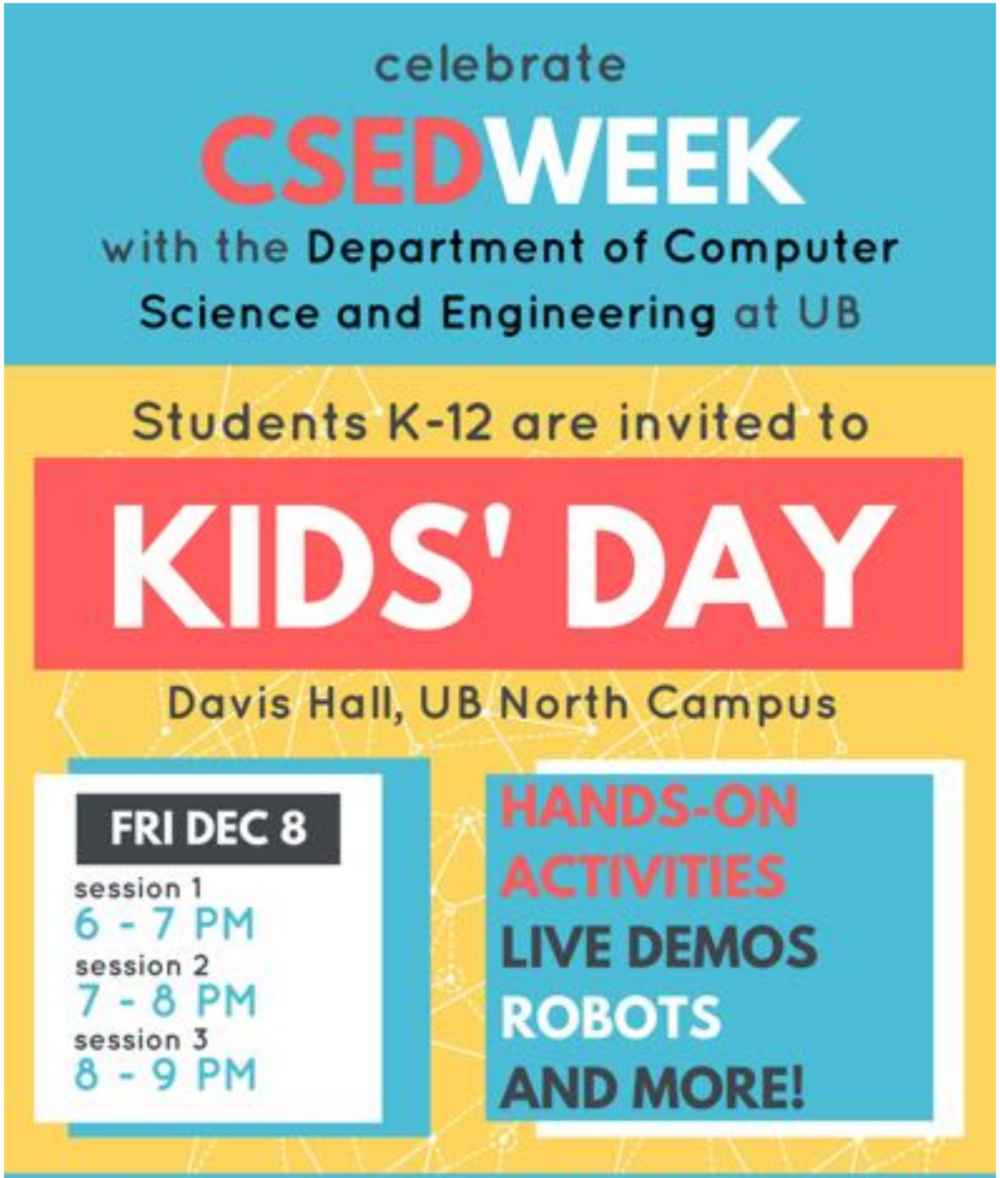
End of the lecture

Graded HW 6

Done by today

Apologies for the delay!

CS Ed week (Dec 8)



SEAS Senior Scholar Program

University at Buffalo
School of Engineering and Applied Sciences

2018 SEAS Senior Scholar Program

The School of Engineering and Applied Sciences (SEAS) Senior Scholar program is an opportunity for undergraduate students at the University at Buffalo to carry out research with a SEAS faculty member. This experience begins during the spring semester of a student's senior year (final year of undergraduate study) and can continue into their graduate program.

Scholarship Information:

- **Master's Degree applicants:** If awarded, students who have applied to a master's degree program will receive a stipend of $100 in Spring 2018. Students will also be eligible to receive another $1,000 if they are accepted and choose to enroll at UB's School of Engineering and Applied Sciences for graduate studies in Fall 2018.

- **PhD Degree applicants:** If awarded, students who have applied to a PhD degree program will receive a stipend of $100 in Spring 2018. Students will also be eligible to receive another $2,000 if they are accepted and choose to enroll at UB's School of Engineering and Applied Sciences for graduate studies in Fall 2018.

Eligibility:

- Currently enrolled UB undergraduate student with Senior standing; expecting to graduate by June 2018

Ask Qs please!

note

stop following | 3 views

Actions

Feedback on Lectures

Continuing on from @807, this post is on the feedback on lectures.

Here are some comments on the feedback on lectures (again most of these were mentioned by more than one person):

1. **More insight into how to come up with the proof/algo in the first place**. Again something that has been on my TODO list but this year, the programming stuff has taken more time than I had anticipated. Will definitely think more on this for next year: I have some ideas (e.g. try to design algos via examples) for next year. If you have any specific suggestion on this, please let me know.
2. **Sometimes get lost in the lecture.** I try to do the best I can but if you do not ask then it is hard for me to gauge if people are lost, or simply do not care or have got everything I said. The more feedback I get, the better I can shape the lectures. I'm perfectly happy to repeat things, go over stuff again, go slower-- whatever it takes so that you understand more of the lecture material. ***But you have to ask questions***: I cannot do this without any feedback. If the only folks who give me feedback are those who understand the material, then I'm not getting the full picture. So please ask if something is not clear! As one comment said, I have done these algorithms and proofs many times, so it is a bit hard for me to figure out which parts are not clear-- so please let me know!

Weighted Interval Scheduling

Input: n jobs (s_i,f_i,v_i)

Output: A schedule S s.t. no two jobs in S have a conflict

Goal: max $\Sigma_{i \text{ in } S}\, v_j$

Assume: jobs are sorted by their finish time

Couple more definitions

$p(j)$ = largest $i < j$ s.t. i does not conflict with j

= 0 if no such i exists

$p(j) < j$

$OPT(j)$ = optimal value on instance $1,..,j$

Property of OPT

j in OPT(j)

j not in OPT(j)

$$OPT(j) = \max \{ v_j + OPT(p(j)), OPT(j-1) \}$$

Given OPT(1),…., OPT(j-1),
how can one figure out if j
in optimal solution or not?

A recursive algorithm

Proof of correctness by induction on j

Compute-Opt(j)

Correct for j=0

If j = 0 then return 0

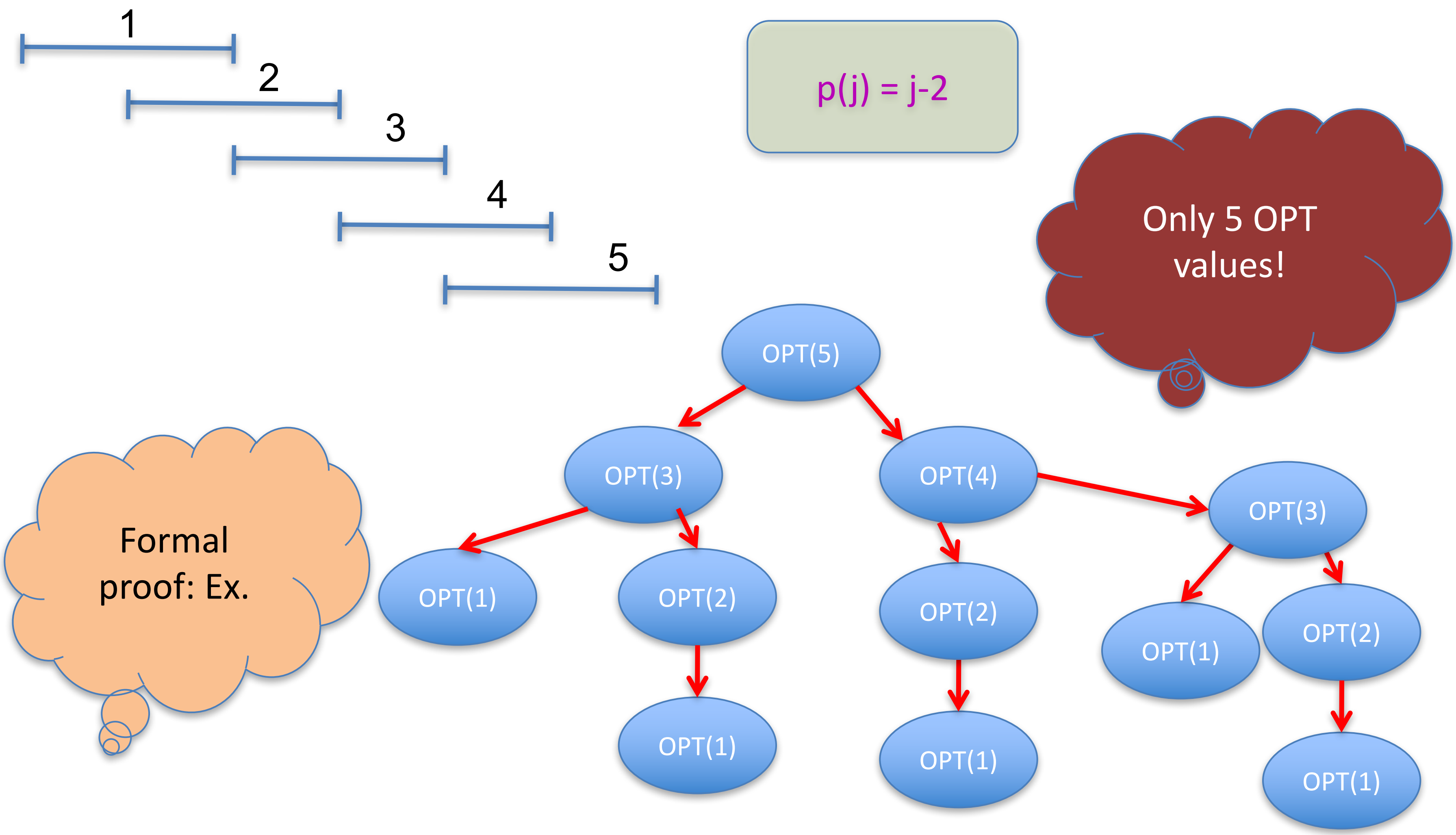
return max { v_j + Compute-Opt(p(j)), Compute-Opt(j-1) }

= OPT(p(j))

= OPT(j-1)

$$OPT(j) = \max \{ v_j + OPT(p(j)), OPT(j-1) \}$$

Exponential Running Time



Using Memory to be smarter

```
Pow (a,n)
    .
    .
    .
    .
  // n is even and ≥ 2
  return Pow(a,n/2) * Pow(a, n/2)
    .
    .
    .
    .
```

O(n) as we recompute!

```
Pow (a,n)
    .
    .
    .
    .
  // n is even and ≥ 2
  t= Pow(a,n/2)
  return t * t
    .
    .
    .
    .
```

O(log n) as we compute only once

How many distinct OPT values?

A recursive algorithm

M-Compute-Opt(j)

If j = 0 then return 0

If M[j] is not null then return M[j]

M[j] = max { v_j + M-Compute-Opt(p(j)), M-Compute-Opt(j-1) }

return M[j]

M-Compute-Opt(j)
= OPT(j)

Run time = O(# recursive calls)

Bounding # recursions

M-Compute-Opt(j)

If j = 0 then return 0

If M[j] is not null then return M[j]

M[j] = max { v_j + M-Compute-Opt(p(j)), M-Compute-Opt(j-1) }

return M[j]

O(n) overall

Whenever a recursive call is made an M value is assigned

At most n values of M can be assigned

Property of OPT

$$OPT(j) = \max\{ v_j + OPT(p(j)), OPT(j-1) \}$$

Given $OPT(1), \ldots, OPT(j-1)$,
one can compute $OPT(j)$

Recursion+ memory = Iteration

Iteratively compute the OPT(j) values

Iterative-Compute-Opt

M[0] = 0

For j=1,…,n

M[j] = max { v_j + M[p(j)], M[j-1] }

M[j] = OPT(j)

O(n) run time

Reading Assignment

Sec 6.1, 6.2 of [KT]

When to use Dynamic Programming

Richard Bellman

There are polynomially many sub-problems

Optimal solution can be computed from solutions to sub-problems

There is an ordering among sub-problem that allows for iterative solution