

CSC 301: Analysis of Algorithms

Week 13: Class wrap up and Final Topics

William Rebelsky
[rebelskyw]

Grinnell College

Monday 6th May, 2024

Table of Contents

- 1 Monday: Dynamic Programming Practice
- 2 Wednesday: Trees and Graphs
- 3 Friday: Linear Programming
 - Linear Programming

CSC 301: Analysis of Algorithms

Class 40: Dynamic Programming and Course Logistics

William Rebelsky
[rebelskyw]

Grinnell College

Monday 6th May, 2024

Reminders

- I have sent preliminary grades to everyone, please review your grades and prioritize resubmitting homeworks and quizzes as needed. Redos are due Friday unless you have asked for and been granted a specific extension.
- Submit any redos that you are able to without worrying about tokens, I will use your tokens in whatever way works best for your final grade
- Other than the Final Project (Due Friday unless extension requested), everything should have been turned in by now
- Class will end 15minutes early today so that you have time to take the end of course survey here:
<https://grinnell.smartevals.com>
- Schedule of the last 3 days:
 - Today: Dynamic Programming and Survey
 - Wednesday: Trees and Graphs Redux
 - Friday: Linear Programming by request

Practice Problems 1

- Consider the problem of neatly printing a paragraph with a monospaced font (all characters having the same width). The input text is a sequence of n words of lengths $l_1, l_2, \dots, l_n$, measured in characters, which are to be printed neatly on a number of lines that hold a maximum of M characters each. No word exceeds the line length, so that $l_i \leq M$ for $i = 1, 2, \dots, n$. The criterion of “neatness” is as follows. If a given line contains words i through j , where $i \leq j$, and exactly one space appears between words, then the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^j l_k$, which must be nonnegative so that the words fit on the line. The goal is to minimize the sum, over all lines except the last, of the cubes of the numbers of extra space characters at the ends of lines. Give a dynamic-programming algorithm to print a paragraph of n words neatly. Analyze the running time and space

Practice Problems 2

- 14.1-2 Show that the following greedy strategy does not work for the rod cutting problem: define the density of a rod of length i as p_i/i i.e. it's value per inch, in each step choose the densest rod possible.
- 14.1-3 Consider a modification to the rod-cutting problem where each cut now costs c instead of 0. Give a dynamic programming solution to this new problem.
- 14.3-4 Show greedy choice fails in Matrix multiplication

Table of Contents

1 Monday: Dynamic Programming Practice

2 Wednesday: Trees and Graphs

3 Friday: Linear Programming

- Linear Programming

CSC 301: Analysis of Algorithms

Class 41: Trees and Graphs Redux

William Rebelsky
[rebelskyw]

Grinnell College

Wednesday 8th May, 2024

Reminders

- I have sent preliminary grades to everyone, please review your grades and prioritize resubmitting homeworks and quizzes as needed. Redos are due Friday unless you have asked for and been granted a specific extension.
- Submit any redos that you are able to without worrying about tokens, I will use your tokens in whatever way works best for your final grade
- Other than the Final Project (Due Friday unless extension requested), everything should have been turned in by now

Red Black Trees Basics

- RBT: BST with color (red or black)
- Keeps trees relatively balanced ($2\lg(n+1)=O(\lg n)$)
- Red Black Properties
 - 1 Every node is either red or black
 - 2 The root is black
 - 3 Every leaf (NIL) is black
 - 4 If a node is red, both its children are black
 - 5 For every node, all simple paths from the node to descendent leaves contain the same number of black nodes. This is the black-height ($bh(x)$) of the node x .
- For convenience, use a single sentinel ($T.NIL$) for all NIL values
- For convenience, we tend to omit drawing the NIL leaves

Minimum Spanning Trees: Kruskal's Algorithm vs Prim's Algorithm

- Kruskal:
 - Keep track of a forest, A , of all vertices in graph G
 - Sort edges in G by weight
 - In each step, choose the lowest weight edge in the graph that connects two distinct components in A
 - Uses Disjoint-set or union-find data structure
- Prim:
 - Keep track of a tree, A , which is the vertices found so far
 - Add light edge between sets $(A, V-A)$
 - Uses a min priority queue of vertices not in the tree

Shortest Path: Bellman-Ford (22.1)

- No negative weight cycles
- Steps:
 - initialize d and π values
 - For i in range of vertices-1: relax all edges
- $O(V^2 + VE)$

Shortest Path: Dijkstra's Algorithm (22.3)

- No negative weight edges
- Faster than Bellman-Ford
- Extended BFS, uses min priority queue keyed by d values instead of FIFO
- Steps:
 - initialize d and π values
 - Insert into the priority Queue Q
 - Extract min value, u, of Q
 - add u to S
 - Relax every edge u,v,w
 - If the call decreased v,d, decrease key(Q,v,v.d)
- Greedy algorithm (choose the closest vertex)

Practice Problems

- 13-3: An AVL tree is a BST that is height balanced: for every node, the heights of its left and right subtrees differ by at most 1. Implemented by recording one additional attribute (height) over a BST.
 - A) Prove that an AVL tree with n nodes has height $O(\lg n)$
 - B) To insert into an AVL tree, insert as usual and then balance the heights. Describe a procedure to perform this balance.
- MST: Run Kruskal's and Prim's algorithms on the graph on the board
- Shortest Paths: Run Bellman-Ford and Dijkstra's algorithms on the graph on the board from points a and f

Table of Contents

- 1 Monday: Dynamic Programming Practice
- 2 Wednesday: Trees and Graphs
- 3 Friday: Linear Programming
 - Linear Programming

CSC 301: Analysis of Algorithms

Class 42: Linear Programming/Last Class

William Rebelsky
[rebelskyw]

Grinnell College

Friday 10th May, 2024

Reminders

- Please Review your grades and work on turning in any outstanding redos as soon as possible. Redos are due Friday unless you have asked for and been granted a specific extension.
- Final Projects are due today. If you need an extension let me know.

Caveats

- This will be an overview only
- Knowledge of this topic will not be evaluated in this course
- I will be going through the material quickly via an example used in the text

Basics

- Goal: Maximize (or minimize) an objective given limited resources and competing constraints
- Requirements: Can define everything in terms of linear functions
- Definitions:
 - Decision Variables: variables that represent choices you can make
 - Constraints: Limits or restrictions
 - Objective: The quantity you would like to maximize or minimize
 - Feasible Solution: satisfies all constraints
 - Infeasible Solution: does not satisfy all constraints
 - Optimal solution: minimum of all feasible solutions
- Many Examples: Scheduling flight crews, deciding what to produce, solving graph problems, single source shortest path

Finding an Optimal Solution

- Optimal solution will occur at a vertex of the feasible region
- Each constraint bounds a half space (cuts the real space in half)
- The intersection of these half spaces is the feasible region and is called a simplex
- Simplex method: Start at vertex and move towards a vertex whose objective value is no smaller. Termination: Local maximum, which must be global max by Duality
- Ellipsoid algorithm and interior point methods: not discussing today
- Integer Linear Programming: all variables must be integers. This is an NP-hard problem

Duality

- Duality: Given a maximization problem (primal), formulate a minimization problem (dual) that has the same objective value. E.g. Max Flow/Min Cut
 - Primal: Let the coefficients in the Objective be c_i , the coefficients in the constraints be a_i , and the constraint maximums be b_i
 - Dual: Then the coefficients in the objective will be the b_i , the coefficients in the constraints will be a_i (Transposed), and the constraint minimums will be c_i

Dual Example

- Primal:

- Maximize $3x_1 + x_2 + 4x_3$ subject to:
- $x_1 + x_2 + 3x_3 \leq 30$
- $2x_1 + 2x_2 + 5x_3 \leq 24$
- $4x_1 + x_2 + 2x_3 \leq 36$
- $x_1, x_2, x_3 \geq 0$

- Dual:

- Minimize $30y_1 + 24y_2 + 36y_3$ subject to
- $y_1 + 2y_2 + 4y_3 \geq 3$
- $y_1 + 2y_2 + y_3 \geq 1$
- $3y_3 + 5y_3 + 2y_3 \geq 4$
- $y_1, y_2, y_3 \geq 0$

Political Problem Description

- You are a politician trying to win an election
- 3 types of voters: urban (100,000), suburban (200,000), rural (50,000). You would like advertise in such a way to have at least half of each of those types of voters vote for you
- Staff research: \$1000 spent on advertising leads to the following thousand vote changes:

Policy	urban	suburban	Rural
Zombie Apocalypse	-2	5	3
Sharks with Lasers	8	2	-5
Highways for flying cars	0	0	10
Dolphins voting	10	0	-2

Table: Information

Political Problem Definitions

- Decision Variables: variables that represent choices you can make:
 - x_1 : Thousands spent on Zombie Apocalypse
 - x_2 : Thousands spent on Sharks with Lasers
 - x_3 : Thousands spent on Highways
 - x_4 : Thousands spent on Dolphins Voting
- Constraints: Limits or restrictions
 - $-2x_1 + 8x_2 + 0x_3 + 10x_4 \geq 50$ (Urban)
 - $5x_1 + 2x_2 \geq 100$ (Suburban)
 - $3x_1 - 5x_2 + 10x_3 - 2x_4 \geq 25$ (Rural)
 - $x_1, x_2, x_3, x_4 \geq 0$ (No negative spending, nonnegativity constraint)
- Objective Function: The quantity you would like to maximize or minimize
 - Minimize $x_1 + x_2 + x_3 + x_4$ subject to the above constraints

Introduce Slack Variables

Minimize $x_1 + x_2 + x_3 + x_4$ subject to

- $-2x_1 + 8x_2 + 0x_3 + 10x_4 - s_1 = 50$ (Urban)
- $5x_1 + 2x_2 - s_2 = 100$ (Suburban)
- $3x_1 - 5x_2 + 10x_3 - 2x_4 - s_3 = 25$ (Rural)
- $x_1, x_2, x_3, x_4 \geq 0$ (No negative spending, nonnegativity constraint)