CS2040S: Data Structures and Algorithms

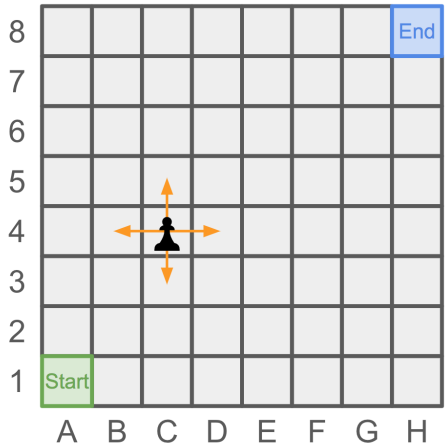
Recitation 8

Goals:

- Advanced graph modelling
- Apply the idea of triangle inequality in shortest paths
- Explore applications of shortest paths problems

Problem 1. Winning the Games

For this problem, consider an $n \times n$ chessboard consisting of n columns and n rows. We have labelled the columns with letters and the rows with numbers. For instance in Figure 1a, your piece is at position C4.



(a)

8	32	44	92	32	14	10	8	5
7	20	54	61	71	18	16	12	32
6	46	80	88	90	9	18	90	7
5	61	83	30	29	22	20	25	10
4	97	50	51	57	30	80	78	15
3	92	89	60	54	88	47	82	33
2	99	83	68	50	23	92	9	68
1	100	76	73	99	55	74	73	22
	A	B	C	D	E	F	G	H

(b)

Figure 1

Every square on the board has a reward associated with it, and your goal is to move a piece from the bottom left corner to the top right corner, while collecting as much points as possible.

The rules of the game are as follows:

- In every move, your piece can move one square in either the vertical or horizontal direction. It cannot move diagonally. For example, in Figure 1a, the piece at square C4 can potentially move to the squares: C5, C3, D4, B4. It cannot move to square D5, D3, B5, or B3.
- Your piece can only move to a square of lesser value. That is, if your piece is on a square with reward r, it can only move to a square with reward $< r$. In Figure 1b above, the piece

at square C4 can move to square C5 and B4 (since $30 < 51$ and $50 < 51$), but not to any other square.

- Your piece starts in square A1.
- When your piece reaches the square in the top right corner of the board (in the example above, H8), it is rewarded with the value of every square it traversed.

Figure 2 below shows an example of a legal set of moves (with a reward of 586).

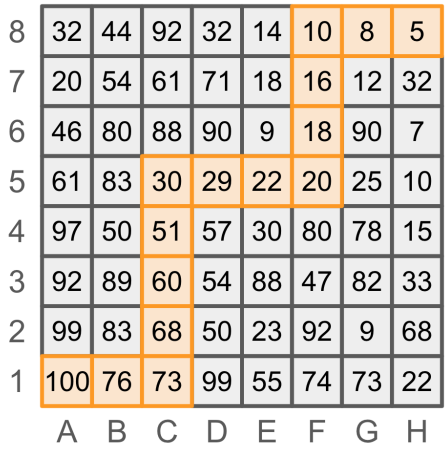


Figure 2

Problem 1.a. Explain how the problem can be modelled as a directed graph. How many nodes does your graph have? State one important property of your graph. Draw a small example (e.g., for $n = 2$ or $n = 3$) to illustrate.

Problem 1.b. Give an efficient algorithm for finding the sequence of moves which attains the maximum possible reward. You may use any algorithm from class in unmodified form without reproducing it. What is the (asymptotic) performance of your algorithm?

Problem 2. StonksX Trader

One day you decided to get into the Foreign Exchange Market (Forex). You registered as a trader in the *StonksX* exchange, a popular exchange for global currencies. There are currently n currencies being traded in the exchange. To help you with analysis, you created a matrix R containing all exchange rates where $R[i, j]$ is the amount of currency j you can get for one unit of currency i. Note that exchange rates are not symmetric: $R[i, j] \neq R[j, i]$. Alas, there is "no arbitrage" possible in the *StonksX* exchange, meaning that if you start with one currency and then convert it to another, and then another, and so on, and then back to the first currency, you will end up with *no more* money then what you started with.

You begin the trading day with an account full of Singapore Dollars (SGD), and you want to end the day with only Great Britain Pounds (GBP). Find the sequence of conversions that will maximize the amount of GBPs you have at the end of the trading day.

Problem 2.a. How do you model this problem as a graph? What are its vertices and edges? Is it weighted or non-weighted? Are the edges directed or non-directed?

Problem 2.b. In your graphical model, which of its property reflects the "no arbitrage" rule? What would happen if such a rule is not enforced by the exchange?

Problem 2.c. What graph algorithm will you use to solve the problem? Which modifications are necessary?

Challenge question: How can you use the algorithm without modifying it (i.e. in the form that was presented to you in lecture)?

Problem 3. Minimum Edit Distance

We have already seen the problem of comparing two strings several times in this module. Another metric for measuring the difference (or equivalently, similarity) between two strings is their *edit distance*. Edit distance measures the *total cost* of transforming from one string to the other via a sequence of character edit operations. In the typical version of this problem, there are three permitted character edit operations:

Operation	Behaviour	Cost
insert(i, c)	Inserts a character c at position i in the string.	$\$ins$
delete(i)	Removes character at position i in the string.	$\$del$
replace(i, c)	Replace a character at position i in the string with character c.	$\$rep$

For example, suppose we are given the following two strings representing nucleotide sequences:

S: `AGGAACCGTA`
T: `AGAATCCGA`

One valid sequence of edits to transform from source string S to target string T is as follows:

1. delete(2): Delete `G` at position 2
2. delete(7): Delete `T` at position 7
3. insert(4, `T`): Insert `T` at position 4

If every edit operation has a cost of 1 (i.e. $\$ins = \$del = \$rep = 1$), then the edit distance due to the sequence of edits above is 1+1+1=3.

Clearly, many possible edit sequences are possible and therefore we are only interested in the *Minimum Edit Distance* (MED): The minimum out of all possible edit distances for transforming a source string to a target string.

Note that the cost for each edit operation varies from problem to problem. When we change the edit operation costs, we may end up with a different MED corresponding to a different sequence of edits.

Problem 3.a. How can we represent the MED between 2 strings as a graph? Let m be the length of source string S and n be the length of target string T, how many vertices and edges does your MED graph have (in terms of m and n)? Draw your MED graph for transforming the string `CG` to the string `AGT`.

Problem 3.b. Suppose that $\$ins = 3$, $\$del = 5$ and $\$rep = 7$. How would you compute the MED between two strings? What graph problem is this? What is the time complexity of your algorithm? Illustrate your solution using the same example of transforming the string `CG` to the string `AGT`.