CSCI 3110 Assignment 1

Instructor: Travis Gagie

posted: 07.05.2020
due: midnight 15.05.2020

Update: questions simplified deadline extended to midnight 19.05.2020

You can work in groups of up to three people. Each group should submit only one copy of the solutions on Brightspace, with all three members' names and banner numbers on it. You may consult with other people but each group should understand the solutions: after discussions with people outside the groups, discard any notes and do something unrelated for half an hour before writing up your solutions. For programming questions you should submit your code, which should compile and run correctly to receive full marks.

This assignment is unusually short in order that it can be completed, marked and returned before the deadline to drop courses without academic penalty. It still seems pretty hard at first, though; the trick is not to give up, break things down into sub-problems, and think about whether you can solve each sub-problem. For example, for Question 3, can you enumerate by brute force all the 3^{18} possible 3-colours of Nova Scotia and, for each one, check if there are two counties sharing a border that are the same colour? Once you've done that, you know how many colourings there are, so as you refine your counting algorithm you can check it's still giving you the same answer. Now, can you count the valid 3-colourings of Cape Breton Island by brute force? Can you count the valid 3-colourings of the mainland by brute force? Their product should be the number of valid 3-colourings of the whole province. Keep going the same way! The scribe notes until 2a are now posted.

There is an optional tutorials on Fridays. There are 4 sections of Collaborate Ultra on Brightspace: if your surname starts A–I join Section 1; J–Q, Section 2; R–W, Section 3; X–Z, Section 4.

1. **(5 marks)** Write an efficient program that reads an integer n between 1 and 100 from the standard input, followed by a newline; reads n strings $S_1, \ldots, S_n$ each of some length ℓ between 1 and 100 and consisting of `A`s, `C`s, `G`s and `T`s, terminated with a newline; prints the largest value k with $1 \leq k \leq \ell$ such that there exists at least one string containing exactly one copy of each distinct k-tuple in $S_1, \ldots, S_n$ and no other k-tuples, followed by a space; and, finally, prints such a string, followed by a newline.

   For example, on the input

```
6
CCAAGATAC
ATTGCCAAG
TGCCAAGAT
AGATACCAG
ACGATTGCC
CCAGAGGAC
```

   your program can print

```
5 ACGATTGCCAAGATACCAGAGGAC
```

   because no string contains every 6-tuple in the given strings and no other 6-tuple, and the returned string contains exactly one copy of each distinct 5-tuple in the given strings and no other 5-tuples.

   Hint: This is basically the same as finding Eulerian tours for the problem on Codeforces (`https://codeforces.com/problemset/problem/508/D`). I apologize for wasting your time with that.

   New hint: For k from 1 to ℓ, extract the *distinct* k-tuples in the input strings and build the de Bruijn graph whose *edges* are those k-tuples. Check if that graph has an Eulerian path. For the example, the de Bruijn graphs whose edges are 3-, 5- and 6-tuples (and thus whose vertices are 2-, 4- and 5-tuples) are shown in Figure 1. Of the three, only the one whose edges are 5-tuples is Eulerian; this is why you check all values of k from 1 to ℓ. You can find a sketch of how to build the graph (badly!) and check for an Eulerian path (also badly!) in the thread "Hint for Question 1" on the discussion group. Adjacency lists would be more efficient here, but "perfect is the enemy of good".

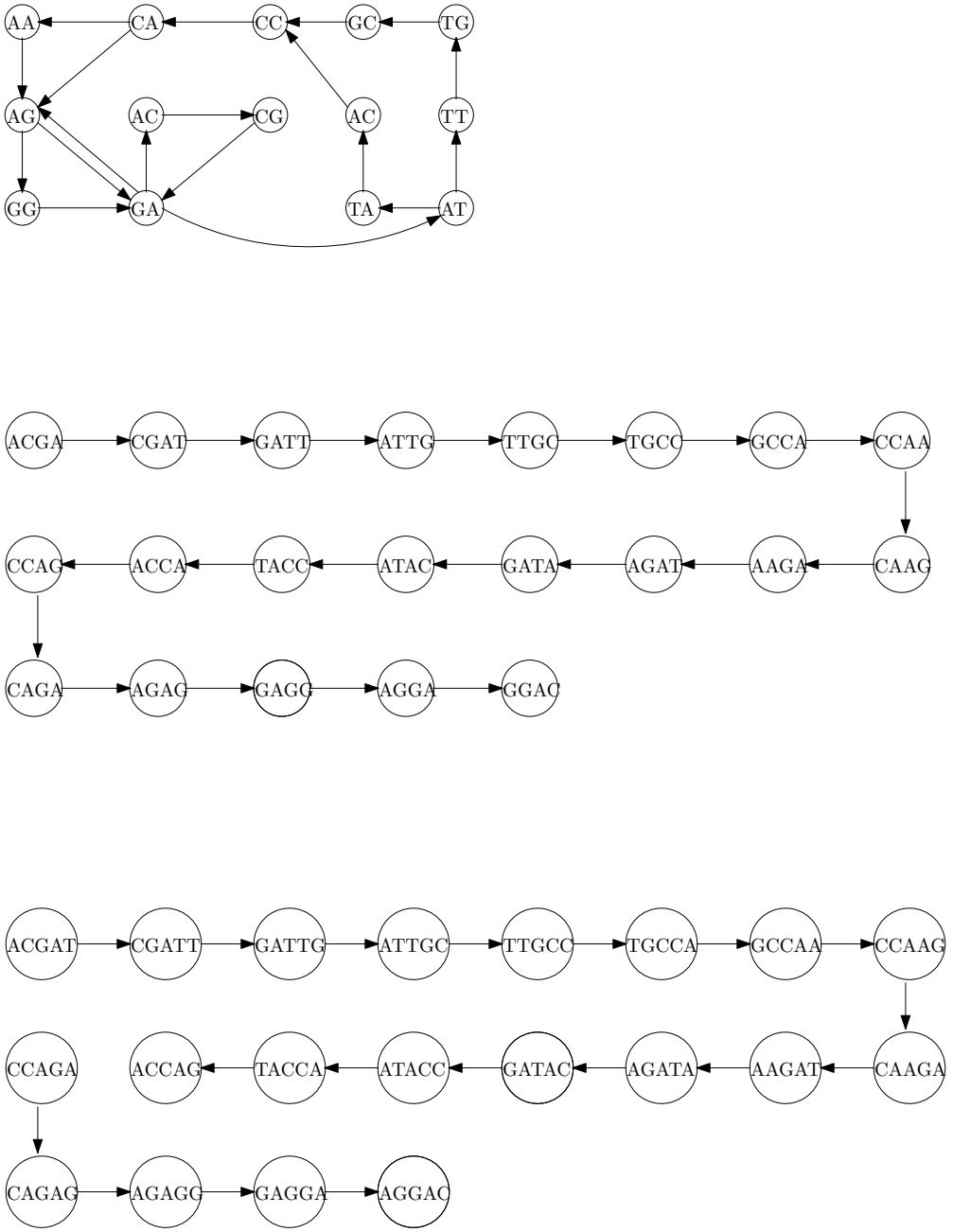


Figure 1: The de Bruijn graphs whose edges are the distinct 3-, 5- and 6-tuples found in the input strings in the example.

2. (a) **(3 marks)** Write a divide-and-conquer program to solve problem 53 on Leetcode (`https://leetcode.com/problems/maximum-subarray/`):

"Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example:

Input: [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6."

Your program should take $O(n \log n)$ time; you do not have to prove that. Ignore the specification on Leetcode that your program take $O(n)$ time, and ignore "**Follow up**". There is a linear-time solution using 1-dimensional dynamic programming but we will not see it until later in the course.

Hint: You can find a discussion of the algorithm on pages 68 to 74 of *Introduction to Algorithms*, which are posted in the assignment folder on Brightspace.

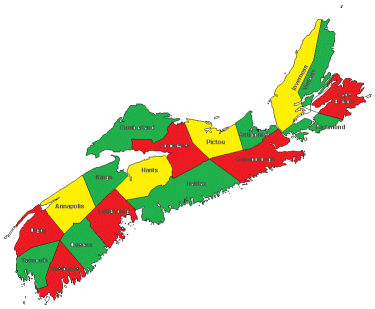
(b) **(2 marks)** Write a program that finds the maximum-sum subarray in a *circular* array — so the subarray can "wrap around" from the end of the array to the beginning — that uses $O(n)$ time plus 2 calls to your program from Part 2a.

Hint: You can find a discussion of the algorithm in the solutions to the fall course's Assignment 1, which are posted in the assignment folder on Brightspace.

New hint: Apparently this is Problem 918 on Leetcode (`https://leetcode.com/problems/maximum-sum-circular-subarray`).

3. **(5 marks)** Write a divide-and-conquer program to count the ways we can colour the counties of Nova Scotia with red, green and yellow such that no two counties that share a border are the same colour. Counties are allowed to be the same colour if they touch only at a point; the map below is an example. Assume that the following lists says which counties touch which.

Annapolis:	Digby, Kings, Lunenburg, Queens
Antigonish:	Guysborough, Pictou
Cape Breton:	Richmond, Victoria
Colchester:	Cumberland, Halifax, Hants, Pictou
Cumberland:	Colchester
Digby:	Annapolis, Queens, Yarmouth
Guysborough:	Antigonish, Halifax, Pictou
Halifax:	Colchester, Guysborough, Hants, Lunenburg
Hants:	Cochester, Halifax, Kings, Lunenburg
Inverness:	Richmond, Victoria
Kings:	Annapolis, Hants, Lunenburg
Lunenburg:	Annapolis, Halifax, Hants, Kings, Queens
Pictou:	Antigonish, Colchester, Guysborough
Queens:	Annapolis, Digby, Lunenburg, Shelburne
Richmond:	Cape Breton, Inverness
Shelburne:	Queens, Yarmouth
Victoria:	Cape Breton, Inverness
Yarmouth:	Digby, Shelburne



Notice that, because Cape Breton Island does not share a border with the mainland, the number of ways to 3-colour Nova Scotia is the number of ways to 3-colour Cape Breton Island times the number of ways to 3-colour the mainland.

Suppose we decide to start counting the 3-colourings of the mainland by counting the ones in which Hants is yellow and Lunenburg is red. Since Hants and Lunenburg together form a *separator* — they split the mainland into relatively small pieces which do not share border — we just need to multiply

- the number of 3-colourings of Kings, Annapolis, Digby, Yarmouth, Shelburne and Queens such that Kings is not yellow or red (because it touches Hants) and Annapolis and Queens are not red (because they touch Lunenburg),
- the number of 3-colourings of Cumberland, Colchester, Pictou, Antigonish, Guysborough and Halifax such that Colchester and Halifax are not yellow (because they touch Hants) and Halifax is not red (because it touches Lunenberg).

Suppose we decide to start counting the allowed 3-colourings of Kings, ..., Queens by counting the ones in which Digby is red and Queens is green. Since Digby and Queens are a separator for Kings, ..., Queens, we just need to multiply

- the number of 3-colourings of Yarmouth and Shelburne such that Yarmouth is not red (because it touches Digby) and Shelburne is not green (because it touches Queens),
- the number of 3-colourings of Kings and Annapolis such that Kings is not not yellow or red (because it touches Hants and Lunenburg) and Annapolis is not red or green (because it touches Lunenburg, Digby and Queens).

It is fairly easy to see that the first number is 3 (Yarmouth is green or yellow and Shelburn is red, or Yarmouth is green and Shelburne is yellow) and the second number is 1 (Kings is green and Annapolis is yellow). Therefore, the number of allowed 3-colourings of Kings, ..., Queens — those in which Kings is not yellow or red, Annapolis is not red, Digby is red and Queens is green — is 6.

How many 3-colourings are there of Nova Scotia in total?

Even just determining whether a map has *any* 3-colourings is NP-hard — we'll talk about this later in the course — so it is unlikely there exists a polynomial-time algorithm to do so. I want you to remember that you can *sometimes* divide and conquer even very hard problems for which you do not expect to find an efficient algorithm, just one that is better than brute force. The key idea in this case is that planar graphs have small separators.

Hint: There's a description of how to do this on the discussion group in the thread "Map Coloring" (except I mistakenly wrote `break` when I meant `continue`), and I sent implementations (now posted in the assignment folder under `Homework-help session` → `Problem3`) that count the colourings using brute force, backtracking, backtracking on Cape Breton Island and the mainland separately. (I also sent an implementation using recursion with no splitting, to give you an idea how you might implement this less kludgily.) In the thread I said how to continue dividing and conquering a bit more, splitting the mainland into two parts. **You should modify the code I sent to do that, and ideally then recurse even one more level**. You're allowed to hard-code in your choices of separators.