CPSC 320: Intermediate Algorithm Design 2019/20 Term 2

Tutorial 4

Prof. Nick Harvey *University of British Columbia*

1. Your friends are planning an expedition to a small town deep in the Canadian north next winter break. They've researched all the travel options and have drawn up a directed graph whose nodes represent intermediate destinations, and edges represent the roads between them.

   They've also learned that extreme weather causes roads in this part of the world to become quite slow in the winter and may cause large travel delays. They've found an excellent travel web site that can accurately predict how fast they'll be able to travel along the roads; however, the speed of travel depends on the time of year. More precisely, the web site answers queries of the following form: given an edge $e = (v, w)$ connecting two sites v and w, and given a proposed starting time t from location v, the site will return a value $f_e(t)$, the predicted arrival time at w. The web site guarantees that:

   - $f_e(t) \geq t$ for all edges e and all times t (you can't travel backwards in time)
   - $f_e(t)$ is a monotone increasing function of t (you do not arrive earlier by starting later).

   Other than that, the function $f_e(t)$ may be arbitrary. For example, in areas where the travel time does not vary with the season, we would have $f_e(t) = t + \ell_e$, where ℓ_e is the time needed to travel from the beginning to the end of edge e.

   Your friends want to use the web site to determine the fastest way to travel through the directed graph from their starting point to their intended destination. (You should assume that they start at time 0, and that all predictions made by the web site are completely correct.) Give an efficient algorithm to do this. (Imagine that a single query to the web site takes a single time step.)

2. Several modern programming languages, including JavaScript, Python, Perl, and Ruby, include a feature called **parallel assignment**, which allows multiple assignment operations to be encoded in a single line of code. For example, the Python code `x,y = 0,1` simultaneously sets x to 0 and y to 1. The values of the right-hand side of the assignment are all determined by the **old** values of the variables. Thus, the Python code $a, b = b, a$ swaps the values of a and b, and the following Python code computes the nth Fibonacci number:

```
def fib(n):
    prev, curr = 1, 0
    while n > 0:
        prev, curr, n = curr, prev+curr, n−1
    return curr
```

   Suppose the interpreter you are writing needs to convert every parallel assignment into an equivalent sequence of individual assignments. For example, the parallel assignment `a,b = 0,1` can be serialized in either order – either `a=0; b=1` or `a=0; b=1` – but the parallel assignment `x,y = x+1,x+y` can only be serialized as `y=x+y; x=x+1`. Serialization may require one (or even more) additional temporary variables. For example, serializing `a,b = b,a` requires a temporary variable.

   Your task is to design an algorithm to determine whether a given parallel assignment can be serialized without additional temporary variables.

   To formalize this, say there are n parallel assignments. The i^{th} assignment is of the form "$v_i \leftarrow R_i$", where v_i is a single variable and R_i is an expression involving many variables.

Simplifying assumptions

- Each variable can only appear on the left-hand side of one assignment.
- Computing a right-hand side R_i doesn't have any "side effects". That is, computing R_i does not involve functions that modify variables.