York University
CSE 3101 Fall 2014 – Unit Test 3 of 5
Instructor: Jeff Edmonds

0 Art therapy question: When half done, draw on the front a picture of how you are feeling.

1. A computational problem is a function P from inputs I to required output $P(I)$. An algorithm is a function A from inputs I to actual output $A(I)$. $Time(A, I)$ gives the running time for algorithm A on input I. Use universal and existential quantifiers (in standard form) to express P is computable time in $3n^2$. How does the prover-adversary game proceed to prove this statement.

   - Answer: $\exists A\ \forall I\ \left[A(I) = P(I)\ and\ Time(A, I) \leq 3|I|^2\right]$
     The prover provides an algorithm A. The adversary provides an input I. The prover wins if A gives the correct answer and uses at most time $3|I|^2$.

2. Consider the following functions.

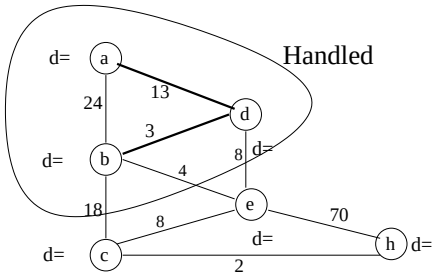
   - **Classification:** Provide a Θ-approximation, if possible classifying the function into $\Theta(1)$, $2^{\Theta(n)}$, $\log^{\Theta(1)}(n)$, or $n^{\Theta(1)}$. What is this class called?
   - **Summation:** Approximate the sum $\Theta(\sum_{i=1}^{n} f(i))$. Which result did you use and how did you use it? What type of sum is it?

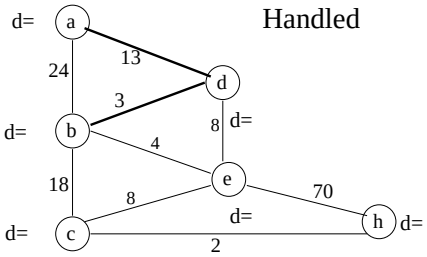
   (a) $f(n) = 5 \cdot n^{5.5} \log^{100}(n)$
   (b) $f(n) = 5 \cdot 3^{4n} n^{100}$
   (c) $f(n) = 5 + sin(n^2)$

   - Answer:
     (a) **Class:** $f(n) = 5 \cdot n^{5.5} \log^{100}(n) = \Theta(n^{5.5} \log^{100}(n))$. For big n, this is bounded between $n^{5.5}$ and $n^{5.6}$. Hence, $f(n) \in n^{\Theta(1)}$. This is a polynomial function.
     Sum: $b^a = 1$ and $d = 5.5 > -1$. Hence, the sum is arithmetic and half the terms are approximately equal. Hence, $\sum_{i=1}^{n} f(i) = \Theta(n \cdot f(n)) = \Theta(n^{6.5} \log^{100}(n))$.
     (b) **Class:** $f(n) = 5 \cdot 3^{4n} n^{100} = \Theta(3^{4n} n^{100})$. For big n, this is bounded between 2^{1n} and 2^{100n}. Hence, $f(n) \in 2^{\Theta(n)}$. This is an exponential function.
     Sum: $b^a = 3^4 = 81 > 1$, the sum is geometric increasing and dominated by the last term. Hence $\sum_{i=1}^{n} f(i) = \sum_{i=1}^{n} 5 \cdot 3^{4i} i^{100} = \Theta(f(n)) = \Theta(3^{4n} n^{100})$.
     (c) **Class:** $f(n) = 5 + sin(n^2)$. The n^2 was a trick because the sine is always between -1 and 1. Hence, $f(n) \in \Theta(1)$. Though it changes with n, the value is always bounded between the constants 4 and 6 and hence is said to be constant.
     Not asked for: This would be the same for $f(n) = \{$ 1 if n is odd, 2 if n is even $\} \in \Theta(1)$.
     Not asked for: If $f(n) = sin(n^2)$, then $f(n)$ would not be in $\Theta(1)$ because things in $\Theta(1)$ should be positive for big n.
     Sum: $b^a = 1$ and $d = 0 > -1$. Hence, the sum is arithmetic and half the terms are approximately equal. Hence, $\sum_{i=1}^{n} f(i) = \Theta(n \cdot f(n)) = \Theta(n)$.

3. Dijkstra's Algorithm:

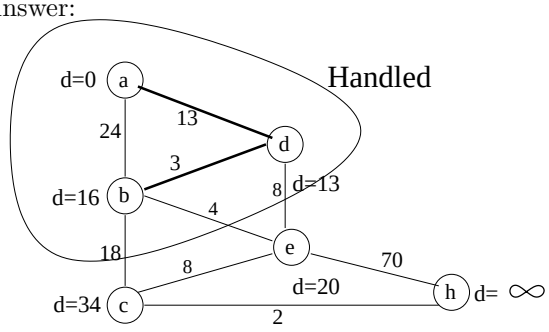
   (a) Consider a computation of Dijkstra's algorithm on the following graph when the circled nodes have been handled. The start node is a. On the left, give the current values of d. (Be sure to think about what the loop invariant says about the values d.)

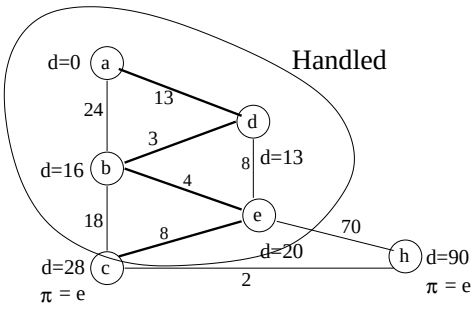




(b) On the right, change the figure to take one step in Dijkstra's algorithm. Include as well any πs that change.

- Answer:





4. Running Time

algorithm $Careful(n)$

⟨***pre−cond***⟩**:** n is an integer.

⟨***post−cond***⟩**:** $Q(n)$ "Hi"s are printed for some odd function Q

```
begin
    if( n ≤ 1 )
        PrintHi(1)
    else
        loop i = 1...n
            PrintHi(i)
        end loop
        loop i = 1...8
            Careful(n/2)
        end loop
    end if
end algorithm
```

algorithm $PrintHi(n)$

⟨***pre−cond***⟩**:** n is an integer.

⟨***post−cond***⟩**:** n^2 "Hi"s are printed

```
begin
    loop i = 1...n^2
        Print("Hi")
    end loop
end algorithm
```

(a) Give and solve the recurrence relation for the number of "Hi"s $Q(n)$. Show your work. Give a sentence or two giving the intuition.

(b) What is the running time (time complexity) of this algorithm as a function of the size of the input. CAREFUL!!!

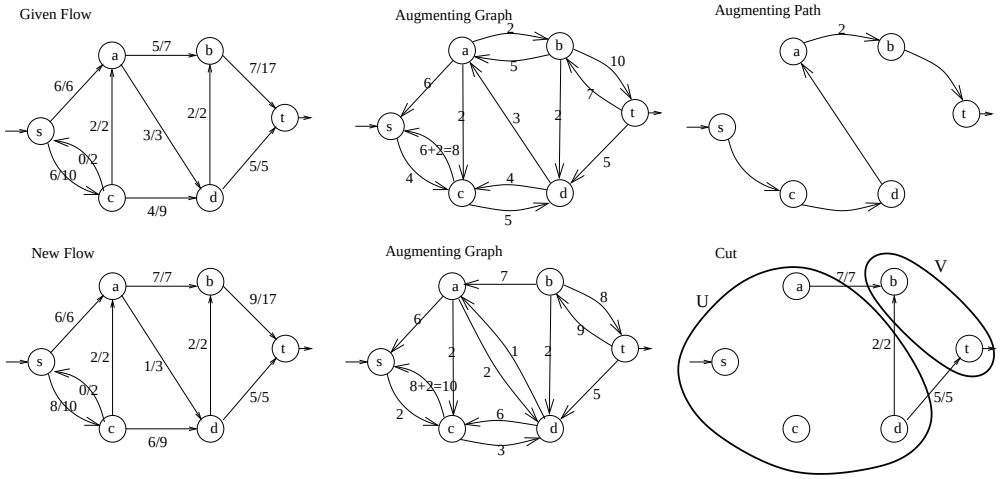
- Answer:
  (a) $Q(n) = 8Q(\frac{n}{2}) + \sum_{i=1}^{n} i^2 = 8Q(\frac{n}{2}) + \Theta(n^3)$. Here $\frac{\log a}{\log b} = \frac{\log 8}{\log 2} = 3 = c$. This means that the $\log n$ levels all have about the same amount of work, giving $Q(n) = \Theta(n^3 \log n)$.
  (b) The size of the input in bits is $size = \log_2(n)$ and the time is $T(size) = \Theta(Q(n)) = \Theta(n^3 \log n) = \Theta((2^s)^3 \log(2^s)) = \Theta(2^{3s}s))$. This is exponential time.

5. Network Flow: Consider the following flow network with source s and terminal (or sink) t. The labels on edges are of the form "f/c" where f is the current flow and c is the capacity of the edge. This flow is feasible.

   (a) Trace one iteration of the network flow algorithm. Use the nodes in the middle fig to give the augmentation graph for this flow. Make the necessary changes to the left figure to give the new flow.

   (b) Prove that there is no flow that is better than this current flow. (Use the right figure if you like.)

   - Answer:



   (Note I am expecting the left two figures in the answer to be drawn on the left most figure on the test sheet, top middle figure drawn on in the middle figure, the bottom middle and the top right figures in the answer are not needed and the bottom right should be given on the right of the test.)
   The rate of this final flow is $f(s,a) + f(s,c) = 6 + 8 = 14$. The cut (U,V) returned by the algorithm has capacity $c(a,b) + c(d,b) + c(d,t) = 7 + 2 + 5 = 14$.
   This flow witnesses that fact that a flow with rate 14 is obtainable.
   This cut witnesses that fact that no flow can be bigger than 14.
   Hence, the rate of the max-flow is 14, giving that this flow is optimum.

   (c) Give an algorithm that solves the Min Cut problem and briefly prove that it works. (A total of approximately four sentences.)

   - Answer: Given an instance $\langle G, s, t\rangle$ run the Max Flow algorithm on it. This returns a flow and a cut with equal values. The cut witnesses the fact that such a small cut can be found. The flow witnesses that fact that there is not a smaller cut or else this flow could not be achieved.

==========

6. (On the exam) NP (Non-Deterministic Polynomial Time)

   (a) What does it mean for a problem P to be in NP?

   - Answer: If *yes* instances I of the problem P have solution/witnesses/proof S which if you had S then you could prove in poly time that your instance is a *yes* instance.
     Given an instance I and a solution S, it is easy to check whether the solution is valid for the instance. The time (and the size of the solution) must be polynomial in the size of the instance.
     An instance is a *yes* instance iff such a solution exists for it.

   (b) What does it mean for a problem P to be NP-Complete.

- Answer: Problem P is in NP and is NP-hard.
  Problem P is NP-hard iff (If given a poly time algorithm for P you can design a poly time algorithm for every problem P' in NP. Note, it is strongly believed that there is not a poly time algorithm for every problem P' in NP. Hence, there is likely not one for P.)