CSC165H1: Problem Set 4 Sample Solutions

Due December 6, 2017 before 10pm

Note: solutions are incomplete, and meant to be used as guidelines only. We encourage you to ask follow-up questions on the course forum or during office hours.

1. **[18 marks] vertex degree** Recall the definition of the degree of a vertex:

Definition 1 (degree of v, $d(v)$). $d(v)$: $|\{(v,u) \mid (v,u) \in E\}|$, for $v \in V, G = (V,E)$

Edit: We deal with simple graphs: the number of vertices is finite and non-zero, there is at most one edge between a given pair of vertices, and there are no edges from a vertex to itself. Although some of the results may extend to graphs with 0 vertices, you don't need to consider these.

(a) **[3 marks]** Prove that for any graph $G = (V,E)$ the number of vertices with odd degree is even.

Solution

Claim: Define $odd(n)$: $\exists k \in \mathbb{Z}, n = 2k+1$, for $n \in \mathbb{Z}$. Then

$$\forall n \in \mathbb{N}, \forall G = (V,E), \neg odd(|\{v \in V \mid odd(d(v))\}|)$$

Proof (induction on $|V|$): Define $P(n) : \forall G = (V,E), \neg odd(|\{v \in V \mid odd(d(v))\}|)$.

base case: If $|V| = 1$ then the number of vertices with odd degree is also zero, an even number. So $P(1)$ is true.

Inductive step: Let $n \in \mathbb{N}$. Assume $P(n)$. I want to show that $P(n+1)$ follows.

Let $G = (V,E)$ be an arbitrary graph with $|V| = n+1$. Since $n+1 > 0$ we can select an arbitrary $v \in V$ and construct $G' = (V', E')$, where $V' = V \setminus \{v\}$ and $E' = E \setminus \{(v,u) | (v,u) \in E\}$.

Since $|V'| = n$, by the IH G' has an even number of vertices of odd degree. Let $i, j \in \mathbb{N}$ be such that $2i$ is the number of vertices in G' with odd degree and j is the number of vertices in G' with odd degree that have an edge to v in G.

An edge to v in G increases the degree of a given vertex by 1 compared to its degree in G'. This means that the j vertices from G' with odd degree and an edge to v in G will have even degree in G, and the $d(v) - j$ vertices in G' with even degree and an edge to v in G will have odd degree. Doing the arithmetic, there will be

$$2i - j + d(v) - j = 2(i-j) + d(v)$$

...vertices with odd degree, except (possibly) v in G. There are two cases to consider.

Case $odd(d(v))$ in G: Then there are $2(j-i) + d(v) + 1$ vertices of odd degree in G, an even number, since the sum of odd numbers is even (by Example 2.18, arithmetic mod 2)

Case $\neg odd(d(v))$ in G: $2(j-i) + d(v)$ vertices of odd degree in G, an even number since the sum of even numbers is even (by Example 2.18, arithmetic mod 2).

in both possible cases $P(n+1)$ follows. ■

(b) **[3 marks]** Let $G = (V,E)$ be a graph with 4 vertices and the following (incomplete) list of degrees:

- $d(v_1) = 1$

- $d(v_2) = 2$
- $d(v_3) = 3$

Provide a complete list of edges that could satisfy the above list of degrees as well as including vertex v_4. Prove that your solution is the only one possible.

Solution

Proof: Let $G = (V, E)$ be a graph with $|V| = 4$ and $V = \{v_1, v_2, v_3, v_4\}$. Assume $d(v_1) = 1, d(v_2) = 2$, and $d(v_3) = 3$. I will show that $E = \{(v_3, v_1), (v_3, v_2), (v_3, v_4), (v_2, v_4)\}$ satisfies the assumed degrees, and it is the only possible set of edges that does so.
Let $E = \{(v_3, v_1), (v_3, v_2), (v_3, v_4), (v_2, v_4)\}$. By counting the number of edges that contain v_1, v_2, and v_3 we verify that these edges satisfy the assumed degrees. To see that this is the only possibility, notice that:

i. Since $d(v_3) = 3$ it must be the case that $\{(v_3), (v_1), (v_3, v_2), (v_3, v_4)\} \subseteq E$.
ii. Since $d(v_2) = 2$ is must have an edge to some vertex other than v_3. Since the degree of v_1 allows only one edge (the one to v_3) that means $(v_2, v_4) \in E$.
iii. Since the previous two observations completely populate v_1, v_2, and v_2 with edges, v_4 has no other vertices to form edges with. ■

(c) **[3 marks]** Prove that every graph $G = (V, E)$ that has at least one vertex v with degree $d(v) = n$ must also have $|V| \geq n + 1$.

Solution

Proof: Let $G = (V, E)$ be a graph and let $n \in \mathbb{N}$. Assume $\exists v \in V, d(v) = n$, and let v be such a vertex. I will prove that $|V| \geq n + 1$.
By the definition of $d(v) = n$ vertex v has n neighbours forming the endpoints of edges incident with v. This means that V contains at least v plus these n neighbours, for $|V| \geq n + 1$. ■

(d) **[3 marks]** Prove that every graph $G = (V, E)$ where $\forall v \in V, d(v) = 2$ has a cycle.

Solution

Proof: Let $G = (V, E)$ be a graph. Let $u \in V$.* Let $v \in V$ where there is a path from u to v that has maximum length among all paths originating at u. Let a maximal path from u to v be $u = v_0, \ldots, v_k = v$.
Since $d(u) = d(v_0) = 2$ we know that u has two neighbours, so any maximal path originating at u has length at least 1. This means that $k \geq 1$ and $v_k \neq v_0$. Since $d(v_k) = 2$, there is an edge from v_k to some vertex other than v_{k-1}. Let v' be a vertex other than v_{k-1} such that $(v_k, v') \in E$. Since there is no longer path originating from u than $v_0, \ldots, v_k$ it follows that v' must duplicate one of the vertices in$\{v_0, v_{k-2}\}$, i.e. $v' = v_j$, where $0 \leq j \wedge j < k - 1$. But then the portion of our original path $v_j, \ldots, v_k$ plus the vertex v_j forms a cycle, since it has at least v_j, v_{k-1}, and v_k distinct vertices. ■

[*] We assume here that $|V| > 0$.

(e) **[3 marks]** Prove that every graph $G = (V, E)$ where $\forall v \in V, d(v) \geq |V| - 3$ and $|V| > 4$ is connected. **Hint:** Assume the graph is not connected and count the vertices adjacent to a pair of disconnected vertices.

Solution

Proof(contrapositive): Let $G = (V, E)$ and assume that G is not connected. I will show it follows that that either $|V| \leq 4$ or V has a vertex v with $d(v) < |V| - 3$. This is equivalent to $\forall v \in V, d(v) \geq |V| - 3 \Rightarrow |V| \leq 4$ (definition of $\Rightarrow$).

Now assume $\forall v \in V, d(v) \geq |V| - 3$. Since G is not connected there are a pair of vertices u and v with no path between them, let u and v be such vertices. Define $N(u)$ as the set of vertices with edges to u and $N(v)$ as the set of vertices with edges to v, and notice that we must have $N(u) \cap N(v) = \emptyset$, since otherwise there would be a path of length 2 between u and v. Adding the two vertices u and v to their neighbourhoods must yield no more than the total of vertices in the graph G, so

$$|V| \geq |N(u)| + |N(v)| + 2 \geq |V| - 3 + |V| - 3 + 2 = 2|V| - 4 \Rightarrow 4 \geq |V| \qquad \blacksquare$$

(f) **[3 marks]** Add the degrees in the previous part and use this sum to calculate the number of edges. Compare your result to Example 6.6 and Example 6.7 from the course notes and explain the connection (or lack thereof).

Solution

discussion: If G has $|V|$ vertices with degree at least $|V| - 3$, then the expression

$$\sum_{v \in V} d(v) \geq |V|(|V| - 3)$$

... counts all the endpoints of edges in G. Since each edge has exactly 2 endpoints, this means that the number of edges that guaranteed that G was connected are at least

$$\frac{|V|(|V| - 3)}{2}.$$

This is smaller than the threshold of $[(|V| - 1)(|V| - 2)/2] + 1$ that was established in Examples 6.6 and 6.7. What gives?

The key observation is that Q1(e) not only specifies how many edges, but how they need to be distributed over vertices, and this allows a lower threshold of edges to guarantee connectedness. For example, the case of a graph made up of an isolated vertex plus the complete graph of the remaining vertices could never satisfy the condition that all vertices have $d(v) \geq |V| - 3$ if $|V| > 4$. $\blacksquare$

2. **[6 marks] binary representation** Read Theorem 4.1, as well as its proof. It is useful to insist that there be a unique (one and only one) binary representation for integers. This can be done by specifying the number of bits in the representation:

(a) **[3 marks]** Prove the following:

For every number $n \in \mathbb{N}^+$, there is a unique representation of n in the following form: $n = \sum_{i=0}^{p} b_i 2^i$, where p is the smallest integer such that $2^{p+1} > n$, p is non-negative, and $b_p, \ldots, b_0 \in \{0,1\}$.

<u>**Solution**</u>

Proof (strong induction on n): Define $P(n)$:

$$\exists p \in \mathbb{N}, \exists b_p, \ldots, b_0 \in \{0,1\}, 2^{p+1} > n \wedge n \geq 2^p \wedge n = \sum_{i=0}^{p} b_i 2^i$$
$$\wedge \left(\exists c_p, \ldots, c_0 \in \{0,1\}, n = \sum_{i=0}^{p} c_i 2^i\right) \Rightarrow \forall i \in \mathbb{N}, i \leq p \Rightarrow c_i = b_i$$

We will prove $\forall n \in \mathbb{N}^+, P(n)$.

base case: $2^{0+1} = 2 > 1$, $2^0 = 1 \leq 1$, and $0 \in \mathbb{N}$. Also $1 = \sum_{i=0}^{0} 1 \times 2^0$ is the only binary representation of 1 of this length (since 0×2^0 gives the wrong sum). so claim $P(1)$ is true.

Inductive step: Let $n \in \mathbb{N}^+$ and assume $\forall k \in \mathbb{N}^+, k \leq n \Rightarrow P(k)$. I will show that $P(n+1)$ follows.

Since $n+1 \geq 2$, there is some integer $1 \leq k \leq n$ such that $n+1 = 2k + b_0'$, where $b_0' = 1$ if $n+1$ is odd, $b_0' = 0$ if $n+1$ is even. By the inductive hypothesis there exists $p \in \mathbb{N}, 2^{p+1} > k \wedge k \geq 2^p$ and $\exists b_p, \ldots, b_0 \in \{0,1\}$ such that $k = \sum_{i=0}^{p} b_i 2^i$, and this is the only binary representation of k for this p.

Let $p' = p+1$, b_0' be as defined above, and $b_{i+1}' = b_i$ for $i = 0, \ldots, p$ and we have

$$n+1 = 2k + b_0' = \left(2\sum_{i=0}^{p} b_i 2^i\right) + b_0' = \left(\sum_{i=0}^{p} b_i 2^{i+1}\right) + b_0' = \sum_{i=0}^{p'} b_i' 2^i$$

To see that this representation is unique, suppose there were another representation, $n+1 = \sum_{i=0}^{p'} b_i'' 2^i$. Use Lemma 4.3 to compare two representations of k:

$$\sum_{i=0}^{p} b_i 2^i = k = \lfloor (n+1)/2 \rfloor = \sum_{i=0}^{p} b_{i+1}'' 2^i$$

By assumption the representation of k using $p+1$ bits is unique, so $b_{i+1}'' = b_i = b_{i+1}'$ for $i = 0, \ldots, p$. Also, since b_0' and b'' are either 0 or 1, depending on whether $n+1$ is even or odd, $b_0'' = b_0'$ so the two representations of $n+1$ are the same.

Also $2^{p+2} = 2 \cdot 2^{p+1} > 2k$, by our assumption, whereas $2^{p+1} = 2 \cdot 2^p \not> 2k$, also by our assumption, so p' is the smallest integer such that $2^{p'+1} > n+1$, and $p' = p+1$ is non-negative because p is ■

(b) **[3 marks]** Use the previous question and some reasoning about 0 to show that there is a unique binary representation of every natural number. Explain why it wasn't just possible to make the domain of the previous proof "every number $n \in \mathbb{N}$"?

Solution

discussion: I can decree that the unique 1-bit binary represention of 0 is $\sum_{i=0}^{0} 0 \times 2^0$. I couldn't simply include this in the previous claim because there is no natural number (or even integer) p such that $2^{p+1} > 0$ and $2^p \leq 0$. The conditions on p, and hence the number of bits, were needed in the previous part to ensure uniqueness by ensuring there were no leading 0s — and any representnation of 0 has a leading 0!
So 0 must be a special case.

3. **[14 marks] algorithm analysis Edit**: Bullet points replaced by an enumerated list below...

(a) **[3 marks]** Prove the claim in Exercise 5.4. The useful formula should be a summation over ir^i rather than ix^i.

Solution

Claim: The average cost of SEARCH on lists containing integers in the open interval $[1, 10]$ is $\Theta(1)$ with respect to n, the length of the lists.

Proof: Let $n \in \mathbb{N}$ and let $\mathcal{I}_n$ be the family of lists consisting of n integers in the open intervaal $[1, 10]$ (duplicates are allowed). There are $10^n = |\mathcal{I}_n|$ such lists, since there are 10 choices for each element of a given list.
Let $x \in \{1, 2, \ldots, 10\}$, and consider the cost of SEARCH finding x over all lists in $\mathcal{I}_n$:

- 1 step to find x in the 10^{n-1} lists that begin with x, for a total cost of $1 \times 9^0 \times 10^{n-1}$
- 2 steps to find x in the $9^1 \times 10^{n-2}$ lists that begin with one of the 9 integers other than x and have x in the second position, for a total cost of $2 \times 9^1 \times 10^{n-2}$
- k steps for find x in the $9^{k-1} \times 10^{n-k}$ lists that have integers other than x in the first $k - 1$ positions and x in the kth position, where k ranges from 1 to n.
- $n+1$ steps to discover that x is not in the list and return an appropriate response for the 9^n lists that have integers other than x in all n positions, for a total cost of $(n + 1)9^n$.

The sum of steps is thus:

$$\begin{aligned}
\sum_{i\in\mathcal{I}_n} RT(i) &= \left(\sum_{k=1}^{n} k9^{k-1}10^{n-k}\right) + (n+1)9^n \\
&= \left[\frac{10^n}{9}\sum_{k=1}^{n} k\left(\frac{9}{10}\right)^k\right] + (n+1)9^n \qquad \text{(factor out constants from sum)} \\
&= \left[\frac{10^n}{9}\sum_{k=0}^{n} k\left(\frac{9}{10}\right)^k\right] + (n+1)9^n \qquad (k=0 \text{ term doesn't change the sum}) \\
&= \left[\frac{10^n}{9}\sum_{k=0}^{n'-1} k\left(\frac{9}{10}\right)^k\right] + (n+1)9^n \qquad (n' = n+1) \\
&= \frac{10^n}{9}\left(\frac{(n+1)(9/10)^{n+1}}{(9/10)-1} + \frac{(9/10)-(9/10)^{n+2}}{((9/10)-1)^2}\right) + (n+1)9^n \\
&\qquad \text{(substituting } n' = n+1 \text{ into formula from 5.4)} \\
&= 10^n\left(10 - \left[\frac{9}{10}\right]^n 9 - (n+1)(9/10)^n\right) + (n+1)9^n
\end{aligned}$$

Dividing through by $|\mathcal{I}_n| = 10^n$ yields an average runtime:

$$\begin{aligned}
\frac{\sum_{i\in\mathcal{I}_n} RT(i)}{|\mathcal{I}_n|} &= 10 - \frac{9^n}{10^n}9 \\
&\approx 10 \in \Theta(1) \qquad (\text{since } \lim_{n\to\infty}(9/10)^n 9 = 0) \qquad \blacksquare
\end{aligned}$$

(b) **[2 marks]** Use precise reasoning to find the average run-time in Exercise 5.4 if the words "1 and 10" are replaced by "1 and 500."

Solution

discussion By substituting 499/500 for 9/10 in the derivation above I get an average runtime of:

$$500 - \frac{499^n}{500^n} \times 499 \approx 500$$

...or $\Theta(1)$ average runtime $\blacksquare$

(c) **[3 marks]** Define the input size of `counter` (below) as $n = s + u$. Find, and prove, a big-Theta bound in terms of n for the worst-case run-time of `counter`.

Solution

Claim: $L(n) = n$ is a lower bound on the run-time of `counter(s, u)` for **all** inputs (including the best-case and worst-case) of size n, and that $U(n) = 6n$ is an upper bound on the run-time of `counter(s, u)` for **all** inputs (including the best-case and worst-case) of size n, so $WC \in \Theta(n)$ and $BC \in \Theta(n)$.

Proof: Let $s, u \in \mathbb{N}$ be inputs for `counter(s, u)`, count the body of the loop as 1 step and note that the maximum decrease of $n = s + u$ in each iteration of the loop is 1, so `counter` requires at least $L(n) = n$ steps for any input of size n, making every input of size n require run-time in $\Omega(n)$

On the other hand, if we let $i \in \mathbb{N}$ be the number of iterations of the loop, tracing the code shows that $n_{i+7} = n_i - 1$. Thus counter requires no more than $7n$ loop iterations, plus 2 steps to account for lines 4 and 12, or $U(n) = 9n$ steps. So every input of size n requires run-time in $\mathcal{O}(n)$

$L(n)$ and $U(n)$ bound **any** input of size n, they are also bounds on the worst case and best case inputs, so $WC \in \Theta(n)$ and $BC \in \Theta(n)$ ■

(d) **[3 marks]** Now, find and prove, a big-Theta bound in terms of n for the **best**-case run-time of `counter`.

Solution

Covered in previous part.

(e) **[3 marks]** Based on the previous two questions, what (if anything) can you say about a big-Theta bound on the average run-time of `counter`? Explain.

Solution

discussion: Let $n \in \mathbb{N}$ and assume $n \geq 6$.[*] Then there are 7 inputs of size n for `counter(s, u)`, i.e. $(s,u) = (n,0), (n-1,1), \ldots, (n-6,6)$. Using $U(n)$ and $L(n)$ from the calculation above, and denoting the number of steps for `counter(s, u)` as $RT(s,u)$ yields an average

$$\frac{\sum_{i=0}^{6} RT(s-i,u+i)}{7} \leq \frac{\sum_{i=0}^{6} U(n)}{7} = \frac{7 \times 9n}{7} = 9n$$
$$\frac{\sum_{i=0}^{6} RT(s-i,u+i)}{7} \geq \frac{\sum_{i=0}^{6} L(n)}{7} = \frac{7 \times n}{7} = n$$

Thus the average runtime is $\Omega(n)$ and $\mathcal{O}(9n)$, or simply $\Theta(n)$ ■

[*] For $n < 6$ the number of valid inputs will be smaller.

```
1  # assume s, u are natural numbers
2  # and that u < 7
3  def counter(s, u):
4      steps = 0
5      while s + u > 0:
6          if u == 0:
7              u = 6
8              s = s - 1
9          else:
10             u = u - 1
11         steps = steps + 1
12     return steps
```