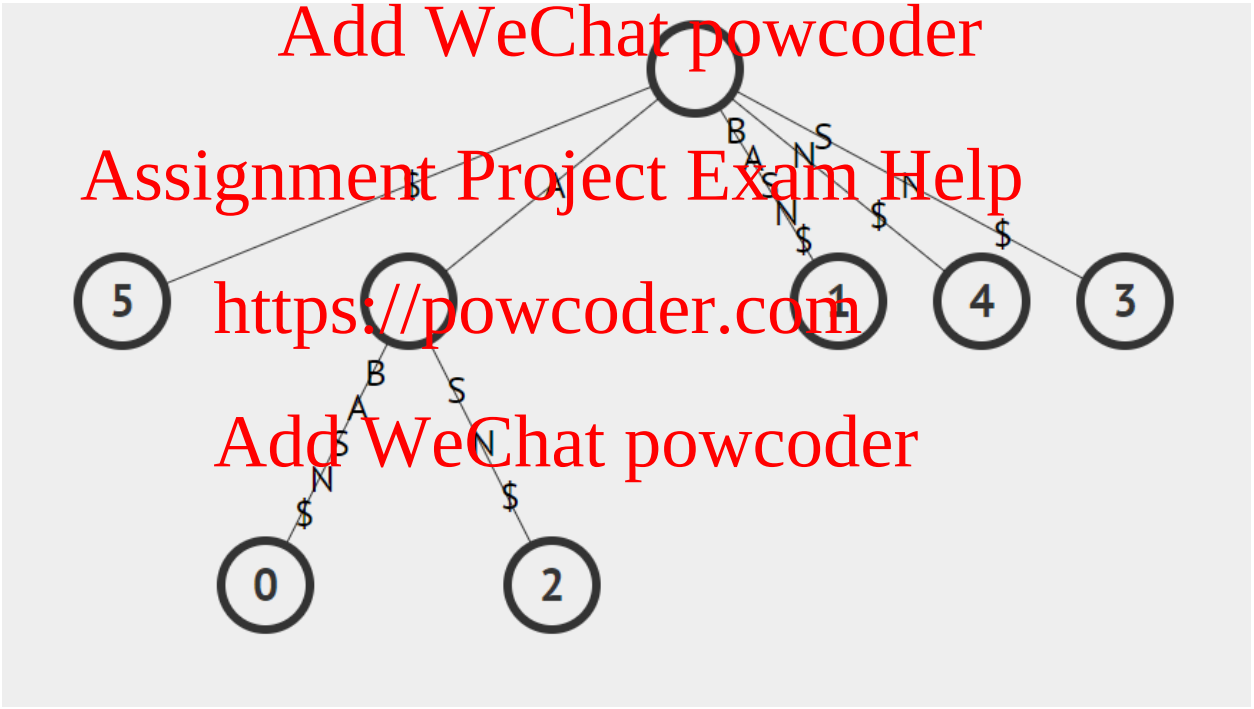
Quiz #4: Solutions

1. Given a suffix tree for string S with suffix link and a string X with length n, the time complexity to find Longest common string between S and X is $O(n \log n)$.

   Solution: The answer is False. The correct time complexity will be $O(n)$. Notice that by using the suffix link you only need to walk through X one time. Check lecture 30 slide 6 for detail explanation

2. Construct the suffix tries for String ABASN$, the constructed tries contains ___ nodes. (include all nodes in the tries beside the root node)

   Solution: The answer is 20. The constructed tries will look like this (in the graph each letter represent a node in tries):



3. The number of children a node have in the suffix tree represent the count of the letter that node represent in the origin string.

   Solution: The answer is False. The number of children and other descendants a node have represent the count of the substring from the root to that node in the origin string. Since we may have many node represent the same letter (letter that edge connect that node in to the upper level represent) in the suffix tree, the statement is incorrect.

4. Given two strings: S and P, what's the time complexity to check if P is a substring of S using suffix tree.

   (a) $O(|P||S|)$
   (b) $O(|P|)$

(c) $\boldsymbol{O(|S|)}$

Solution: $O(|P||S|)$ is the time complexity of the brute force method. To solve this problem by suffix tree, we first construct the suffix tree of S which takes $O(|S|)$, and then we check if P is a substring of S by iterating the characters in P in the suffix tree of S until either we find it is a substring or not. Note that if P is longer than S then we don't need to iterate all characters in P.

5. Given two strings: S_1 and S_2, what's the time complexity to find the longest common substring of S_1 and S_2 using suffix tree.

(a) $O(|S_1||S_2|)$

(b) $O(|S_1|^2|S_2|)$

(c) $O(|S_1||S_2|^2)$

(d) $\boldsymbol{O(|S_1|+|S_2|)}$

Solution: To solve this problem by suffix tree, we can construct the suffix tree of $S_1\#S_2\$$, where a node with both # and $ as its descendants represents a substring of S_1 and S_2, and we need to iterate all the nodes to find the longest one. The overall time complexity is $O(|S_1|+|S_2|)$.

6. Which one is the major drawback of suffix trees:

Solution:

(a) Need preprocessing. This description is true, but this is not a major drawback. Most data structure requires a preprocessing and an efficient and powerful preprocessing is part of the essence of a data structure.

(b) Non-optimal space complexity. This is not true. The linear complexity is optimal, because we have to at least store the string in linear space. Otherwise, we can't 'distinguish' some two different strings when the length is large.

(c) Large time complexity of answering common string queries. This is not true. Most queries is answered in the optimal runtime. Again, we have to at least read in the whole query, e.g., two indices ($O(1)$-time) or one query string ($O(|\text{query}|)$-time).

(d) **Require a lot of memory/space**. This is discussed in one of the lectures. People have developed many methods to address this problem, but the current algorithms are not satisfactory enough.

(e) Can efficiently handle only a few types of string queries. This is not true. (Generalized) suffix trees are very very powerful and efficient.

7. Assume algorithm A is a randomized algorithm with expected running time (wall time) $f(n) = 10n$, and algorithm B is deterministic and has running time $g(n) = 2n \log n$. As $n \to +\infty$, the probability that algorithm A is slower than algorithm B does not necessarily vanish (approach to zero).

Solution: False, the probability vanishes for any f and g such that $f(n) = o(g(n))$.

Let $f(n, s)$ be the wall time when the random numbers are s, e.g., the coin-flip results used in the skip list. By definition, we have $\mathbb{E}_s[f(n, s)] = f(n)$. By the Markov's inequality, we know

$$\Pr[f(n, s) \geq g(n)] \leq \frac{f(n)}{g(n)}$$

As n approaches $+\infty$, the right-hand-side vanishes. Therefore, by the squeeze theorem, the probability also vanishes.