CSE 21 Fall 2018

Homework 6

Due: Sunday, November 11, 2018 at 10:00pm

Instructions

Homework should be done in groups of **one to three** people. You are free to change group members at any time throughout the quarter. Problems should be solved together, not divided up between partners. Homework must be submitted through **Gradescope** by **a single representative**. Submissions must be received by **10:00pm** on the due date, and there are no exceptions to this rule.

You will be able to look at your scanned work before submitting it. Please ensure that your submission is **legible** (neatly written and not too faint) or your homework may not be graded.

Students should consult their textbook, class notes, lecture slides, instructors, TAs, and tutors when they need help with homework. Students should not look for answers to homework problems in other texts or sources, including the Internet. You may ask questions about the homework in office hours, but **not on Piazza**.

Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should **always explain** how you arrived at your conclusions and **justify your answers** with mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to **convince the reader** that your results and methods are sound.

For questions that require pseudocode, you can follow the same format as the textbook, or you can write pseudocode in your own style, as long as you specify what your notation means. For example, are you using "=" to mean assignment or to check equality? You are welcome to use any algorithm from class as a subroutine in your pseudocode. For example, if you want to sort list A using InsertionSort, you can call InsertionSort(A) instead of writing out the pseudocode for InsertionSort.

Required Reading Rosen 10.4 through Theorem 1, 11.1, 11.2 through Theorem 2

Key Concepts: DAGs and topological orderings, graph search and reachability, rooted and unrooted trees, basic counting principles (sum and product rules)

---

1. A daily flight schedule is a list of all the flights taking place that day. In a daily flight schedule, each flight F_i has an origin city OC_i, a destination city DC_i, a departure time d_i and an arrival time $a_i > d_i$. This is an example of a daily flight schedule for November 11, 2018, listing flights as $F_i = (OC_i, DC_i, d_i, a_i)$:

$$
\begin{aligned}
F_1 &= (\text{Portland, Los Angeles, 7:00am, 9:00am})\\
F_2 &= (\text{Portland, Seattle, 8:00am, 9:00am})\\
F_3 &= (\text{Los Angeles, San Francisco, 8:00am, 9:30am})\\
F_4 &= (\text{Seattle, Los Angeles, 9:30am, 11:30am})\\
F_5 &= (\text{Los Angeles, San Francisco, 12:00pm, 1:00pm})\\
F_6 &= (\text{San Francisco, Portland, 1:30pm, 3:00pm})
\end{aligned}
$$

(a) (2 points) Given any daily flight schedule, describe how to construct a DAG so that paths in the DAG represent possible sequences of connecting flights a person could take. What are the vertices, and when are two vertices connected with an edge?

Solution: The vertices are the flights in the daily flight schedule. Draw a directed edge from flight F_i to flight F_j if

- $DC_i = OC_j$ (flight F_j leaves from where flight F_i lands), and
- $a_i \leq d_j$ (flight F_j leaves after flight F_i lands).

(b) (2 points) Why is your graph always a DAG?

Solution: Our graph is always a DAG because it's clearly a directed graph, and a cycle would be impossible since that would require you to take the same flight more than once on a given day. This is impossible because once you have taken a flight, some time has elapsed, and you cannot go back in time to take the same flight again.

More formally, we can say that if there is cycle involving F_i and F_j then there must be a directed path from F_i to F_j and from F_j to F_i. Then according to our definition of edges, a directed path from F_i to F_j means $a_i \leq d_j$ and $a_j \leq d_i$. Also, by definition, $a_i > d_i$ and $a_j > d_j$. But then putting these together we have $a_i \leq d_j < a_j \leq d_i$ which is a contradiction to the fact that $a_i > d_i$.

(c) (2 points) What problem would you need to solve in your DAG to help you determine the maximum number of flights a person could take on a given day?

Solution: Since paths in the DAG represent possible sequences of connecting flights, the maximum number of flights a person could take is determined by the longest path in the graph.

(d) (2 points) Draw the DAG you described for the given example of November 11, 2018 and give the maximum number of flights a person could take on that day.

Solution:

$$\begin{tikzcd}
 & F_1 \arrow[dd] & F_2 \arrow[dd] & \\
F_6 & & & F_3 \arrow[lll] \\
 & F_5 \arrow[ul] & F_4 \arrow[l] &
\end{tikzcd}$$

Using the graph shown above, we see that the longest path includes four flights (F_2, F_4, F_5, F_6), so the maximum number of flights a person could take on November 11, 2018 is four.

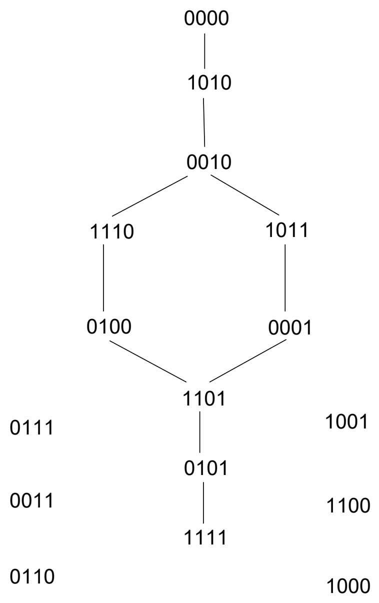
2. In this puzzle, a traveler needs to transport a lion, a poodle, and a giant chocolate bar across a river, but his boat can only fit himself plus one of the other three. A configuration of this puzzle can be described as a length 4 binary string, where each bit from left to right describes the status of the traveler, the lion, the poodle, and the chocolate, respectively. A 0 means that the item is on one side of the river (the starting side) and a 1 means that the item is on the other side of the river. For example the configuration 1100 means that the traveler and the lion are on not on the starting side of the river, and the poodle and the chocolate are on the starting side of the river.

   (a) (4 points) Which of the 16 possible configurations can actually be reached from the starting configuration (0000) if the following rules apply?

   - It is forbidden to leave the lion and the poodle on a shore together without the traveler, or the lion will eat the poodle.
   - It is forbidden to leave the poodle and the chocolate on a shore together without the traveler, or the poodle will eat the chocolate.

   Justify your answer using an appropriate graph.

   Solution:



   From this graph, we can see that 10 of the 16 possible configurations are reachable from 0000. These configurations are:

0000	1010	0010	1110	1011
0100	0001	1101	0101	1111

   (b) (4 points) How can the traveler, the lion, the poodle, and the chocolate all make it across the river in the fewest number of boat trips possible? Say how many boat trips are required, and justify why this is the minimum amount by referring to the graph used in part (a).

   Solution: The puzzle can be solved in a minimum of seven boat trips, because the shortest path from 0000 to 1111 includes seven edges. There are actually two such solutions using this minimum amount of seven trips. One solution is:

1. The traveler crosses with the poodle. (0000 to 1010)
2. The traveler crosses alone. (1010 to 0010)
3. The traveler crosses with the lion. (0010 to 1110)
4. The traveler crosses with the poodle. (1110 to 0100)
5. The traveler crosses with the chocolate. (0100 to 1101)
6. The traveler crosses alone. (1101 to 0101)
7. The traveler crosses with the poodle. (0101 to 1111)

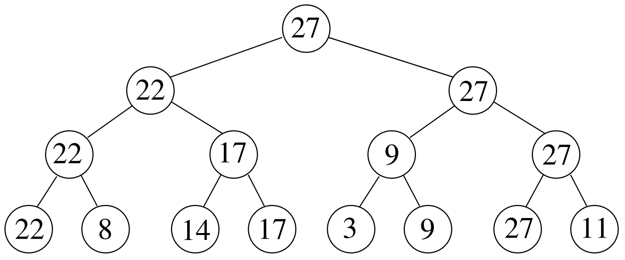
The other solution is:

1. The traveler crosses with the poodle. (0000 to 1010)
2. The traveler crosses alone. (1010 to 0010)
3. The traveler crosses with the chocolate. (0010 to 1011)
4. The traveler crosses with the poodle. (1011 to 0001)
5. The traveler crosses with the lion. (0001 to 1101)
6. The traveler crosses alone. (1101 to 0101)
7. The traveler crosses with the poodle. (0101 to 1111)

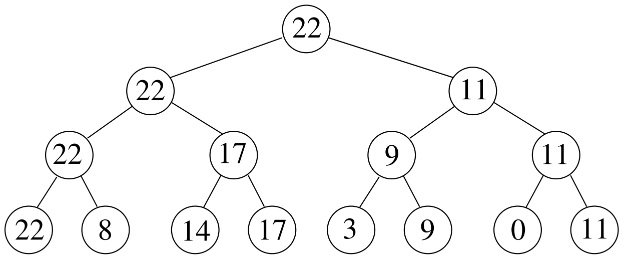
(c) (3 points) Now suppose that the traveler must pay a toll of one dollar every time he crosses the river with an animal. How can they all make it across the river while paying the least money possible? Say how much money is required, and justify why this is the minimum amount by referring to the graph used in part (a).

Solution: Each of the two shortest-path solutions given in part (b) requires four dollars. Any other path would require at least as many trips and at least as much money, so either of the solutions above will be optimal, using the minimum amount of money, four dollars.

3. A sorting algorithm that uses a binary tree to sort a list of positive integers $a_1, a_2, \dots, a_n$ from largest to smallest can be described as follows. In the first step, we construct the binary tree. The elements $a_1, a_2, \dots, a_n$ are the leaves of the tree, and we build up the tree one level at a time from there. From left to right, compare the elements in pairs, and put the larger of the two as the parent vertex. Do this at each level until reaching the root, which will be the largest element. Output the value at the root. For example, if the list to be sorted is $22, 8, 14, 17, 3, 9, 27, 11$, the tree would look like this:



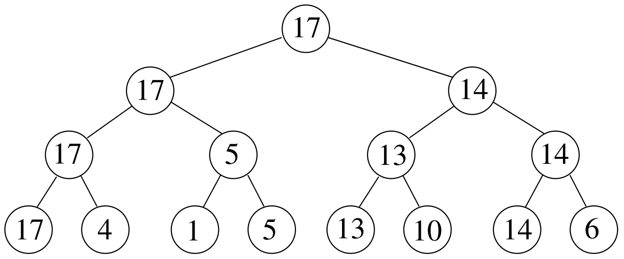
Then, in the second step, remove the leaf corresponding to that largest element, and replace it with a leaf labeled 0, which is defined to be smaller than all the other list elements. Recompute the labels of all vertices on the path from this 0 to the root. That is, relabel all vertices on the path from the 0 to the root by choosing the larger of the values of their two children. Then the root will become the second-largest element. Output the value at the root. In our example, the tree would now look like this:



We can repeat this procedure a total of n times, which will be enough to output the entire list in decreasing order.

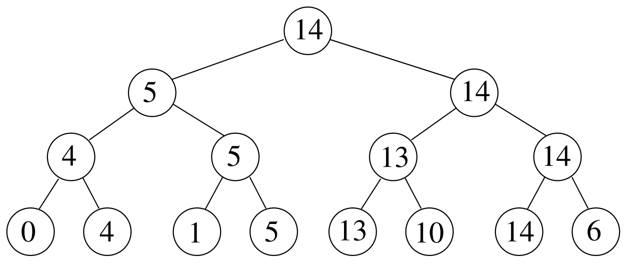
(a) (4 points) Trace through the algorithm as applied to the list $17, 4, 1, 5, 13, 10, 14, 6$. Show the tree at each step.

Step 1:



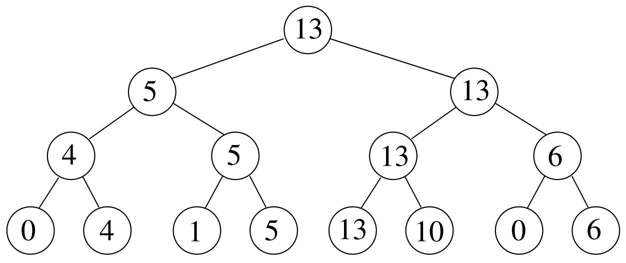
Output 17.

Step 2:



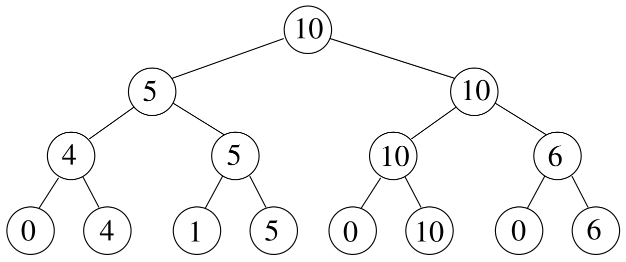
Output 14.

Step 3:



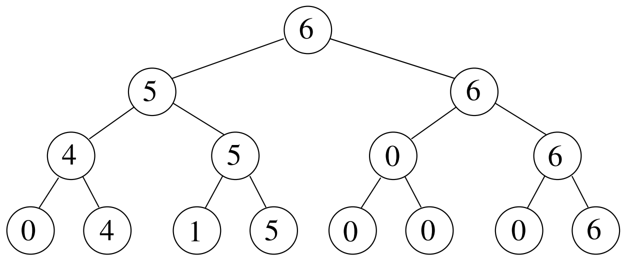
Output 13.

Step 4:



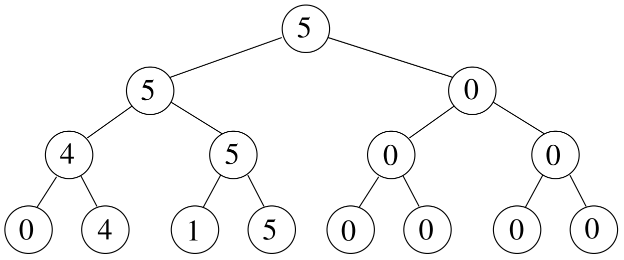
Output 10.

Step 5:



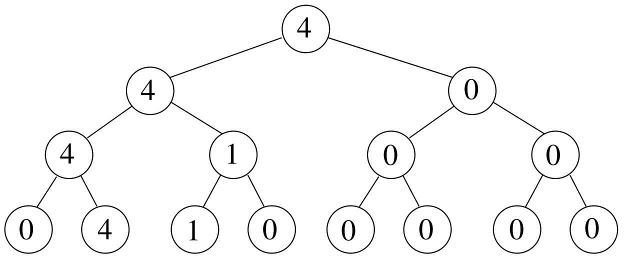
Output 6.

Step 6:



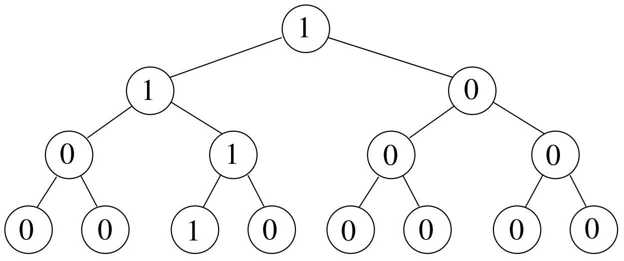
Output 5.

Step 7:



Output 4.

Step 8:



Output 1.

(b) (3 points) If n is a power of two, as in the example, how many comparisons are done at each step?

Solution: In the first step, we need to compute the label of all internal vertices of the tree. If there are n leaves, there are $n-1$ internal vertices, so $n-1$ comparisons are done in the first step. At each subsequent step, we only need to recompute the labels of all vertices on a path from the leaves to the root. There is one comparison at each level (excluding the lowest level with leaves), which means we do $\log_2 n$ comparisons at each step besides the first step.

(c) (3 points) If n is a power of two, as in the example, how many comparisons are done throughout the entire algorithm? What is the order of the algorithm in Θ notation?

Solution: We know from part (b) that we do $n-1$ comparisons in the first step, and $\log_2 n$ comparisons at each of the other $n-1$ steps. Therefore, the total number of comparisons throughout the entire algorithm is $(n-1)+(n-1)\log_2 n = n\log_2 n + n - \log_2 n - 1$. In Θ notation, this is $\Theta(n\log_2 n)$ from taking the largest of the individual terms.

4. Huffman code is a way to encode information using **variable-length binary strings** to represent symbols depending on the frequency of each individual letter. Specifically, letters that appear more frequently can be encoded into strings of shorter lengths, while rarer letters can be turned into longer binary strings. On average, Huffman code is a more efficient way to encode a message as the number of bits in the output string will be shorter than if a fixed-length code was used.

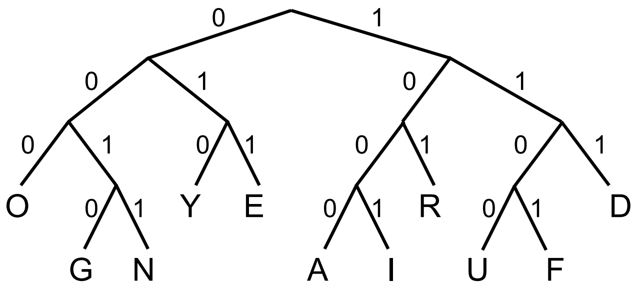
   In addition, this encoding scheme is proven to be unambiguous in the sense that we can easily identify the boundaries between letters and uniquely decrypt an encoded message. That is, no letter's encoding can be the prefix of another letter's encoding (e.g., we cannot have both 00 and 001 as the encoding for two different letters).

   Huffman encoding is managed through a full binary tree, called the **Huffman tree.** Here, the leaf vertices are letters in the original message. For each internal vertices, the left outgoing edges (branches) are labeled with a 0, and right branches are labeled with a 1. The path from the root to the leaf gives us encoding of the corresponding letter in the leaf

   (a.) (2 points) An encrypted file contains the following sequence of binary digits

   0100001100100010101111100010010011001011011001001101 1

   Decode the message knowing it was encoded using the Huffman code given by the tree below



   Solution: We read the encoded message from left to right as we move along the binary tree. Once you reach the end of a path (i.e. a leaf), remove the current segment in the binary string and replace it with the symbol of the leaf.

010	000	1100	1000	101	011
Y	*O*	*U*	*A*	*R*	*E*

111	000	1001	0011	0010	1101	1001	0011	011
D	*O*	*I*	*N*	*G*	*F*	*I*	*N*	*E*

   Thus, the original message is "YOUAREDOINGFINE" or "YOU ARE DOING FINE"

   (b.) (3 points) Suppose a certain file contains only the following letters with the corresponding frequencies

A	B	C	D	E	F	G
73	9	30	44	130	28	16

   In a fixed-length encoding scheme, each character is given a binary representation with the same number of bits. What is the minimum number of bits required to represent each letter of this file under fixed-length encoding scheme? Describe how to encode all seven letters in this file using the number of bits you gave earlier. What is the length of the encoded file under this encoding scheme?

Solution: Since a fixed-length encoding scheme of k bits gives us n unique binary string to use for the letters, where $n = 2^k$. So given 7 characters in the file, the number of bits needed is given by $\lceil \log_2(7) \rceil = 3$.

Here is an example for encoding seven letters using fixed-length encoding scheme: $A = 000, B = 001, C = 010, D = 011, E = 100, F = 101, G = 110$.

The length of the encoded file is

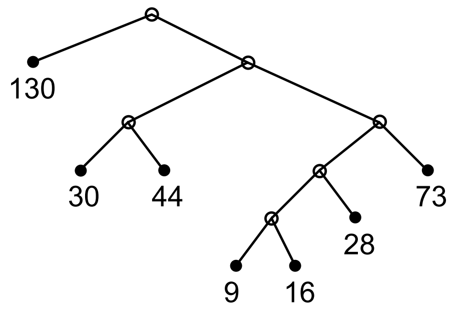
$$3 * (73 + 9 + 30 + 44 + 130 + 28 + 16) = 990 \text{ bits}$$

(c.) (5 points) The following is the algorithm for building Huffman tree (Rosen p.764)

```
procedure Huffman(C: letters $a_i$ with frequencies $w_i$ , $i = 1, \ldots, n$)
1.  $F$ := forest of $n$ rooted trees, each consisting of the single vertex $a_i$ and weight $w_i$
2.  while $F$ is not a tree
3.      Replace the rooted trees $T$ and $T'$ of least weights from $F$ with $w(T) \leq w(T')$
        with a tree having a new root that has $T$ as its left subtree and $T'$ as its right
        subtree.
4.      Label the new edge to $T$ with 0 and the new edge to $T'$ with 1
5.      Assign $w(T) + w(T')$ as the weight of the new tree.
{the Huffman coding for the symbol $a_i$ is the concatenation of the labels of the edges in
the unique path from the root to the vertex $a_i$}
```

Construct the Huffman code that enables you to encrypt the same file in part (b) so that you can store it using the least number of bits. Specify the binary representation for each letter and compute the length of the encoded file under Huffman coding

Solution: The binary tree for this Huffman code is given by



Under this Huffman tree, the codeword for each letter is

A	B	C	D	E	F	G
111	11000	100	101	0	1101	11001

The length of the encoded file is

$$3 * 73 + 5 * 9 + 3 * 30 + 3 * 44 + 1 * 130 + 4 * 28 + 5 * 16 = 808 \text{ bits}$$

5. Give an exact answer for each part, and explain the reason for your answer.

(a.) (4 points) How many four-digit positive integers are there in which all the digits are different?

Solution:

- There are 9 ways to choose the first digit (any of the 10 digits except 0)
- For each of those choice, there are 9 ways to choose the second digit (any of the 10 digits except out choice for the first)
- There are 8 ways to choose the third digit
- There are 7 ways to choose the last digit

Hence, the required number is $9 * 9 * 8 * 7 = 4536$

(b.) (4 points) How many bitstrings of length ten are made up of five zeros and five ones?

Solution: We just need to choose five of the ten possible positions to place the zeros, say, and then the binary string is completely determined. Therefore, the number of bitstrings of length ten with five zeros and five ones is $\binom{10}{5} = 252$.

(c.) (4 points) How many strings of length ten consist of two different digits alternating? For example, 4747474747.

Solution: We have 10 options for the first digit of the string, and 9 options for the second digit of the string, since the two digits must be different. These two choices determine the rest of the string, so the number of possible strings is $10 * 9 = 90$.

(d.) (4 points) For about \$25, you can build your own lightsaber at Disneyland and fulfill you dream of becoming a Jedi Master or a Sith Lord. A lightsaber must include an emitter, a blade, and a hilt. Suppose you get to choose from 5 types of emitters, 4 types of blade colors, and 2 types of hilts. In addition, you also have the option of adding a switch, an accessory ring, a customized hilt sleeve, an adapter. How many way are there to build your lightsaber in total?

Solution: We use the product rule and multiply the number of options for items in the lightsaber. For the optional items, there are 2 options each (either switch or no switch, etc.). This gives $(5 * 4 * 2) * (2 * 2 * 2 * 2) = 640$.

(e.) (4 point) A California license plate follows the format $DLLLDDD$, where D represents a digit and L represents an upper case letter. For example, a valid California license plate is $5JBC434$ (my plate). How many California license plates are possible?

Solution: There are 26 upper case letters and 10 digits. Using the Product Rule and the number of options in each position gives $10 * 26 * 26 * 26 * 10 * 10 * 10 = 175760000$.

(f.) (4 points) How many bits of memory does the California DMV need to allocate in its database to store each license plate number?

Solution: The number of bits needed to store each object when there are n objects is $\lceil \log_2 n \rceil$. So the number of bits of memory needed to store each license plate is $\lceil \log_2 175760000 \rceil = \lceil 27.389 \rceil = 28$.

(g.) (4 points) How many rearrangements are there of the word TENNESSEE?

Solution: Here, we are trying to rearrange the letters in an alphabet with $1T, 2N, 2S, 4E$. Since there are repeated letters, we shall put subscripts on the repeated letters, so that we can tell them apart. The number of ways to rearrange $T, N_1, N_2, S_1, S_2, E_1, E_2, E_3, E_4$ is the number of permutations of nine objects, which is $9!$.

Now we will group some of these subscripted strings into the same category if they result in the same word when we remove the subscripts. We are free to swap the positions of each of the two ones, each of the two threes, and so on, without changing the underlying word. Since there are 4! ways to swap the E's, 4! ways to swap the S's, and 4! ways to swap the N's, using the categories formula, the number of categories (words) comes from the number of objects (9! subscripted strings) divided by the size of each category $(4! * 2! * 2!)$. This gives $\frac{9!}{4!2!2!1!} = 3780$ rearrangements of the word TENNESSEE.

(h.) (4 points) How many of the rearrangements in part (a.) do not have the two S's next to each other?

Solution: We first remove all the S's from the word $TENNESSEE$ and rearrange the remaining letters $TENNEEE$. Similar to the argument in the previous part, there are $\frac{7!}{4!2!1!}$ ways to do that.

Now for each arrangement, we shall attempt to insert the two S's back to the word. Observe that for each arrangement, there are 8 possible spots to place the S's - either at the beginning, at the end, or between ay two letters (see an example below)

$$-N-E-T-E-N-E-E-$$

Thus, there are $\binom{8}{2}$ ways to place the two S's in each rearrangement.

In total, the number of valid rearrangements are $\binom{8}{2}\binom{7}{4,2,1} = 2940$ rearrangements.