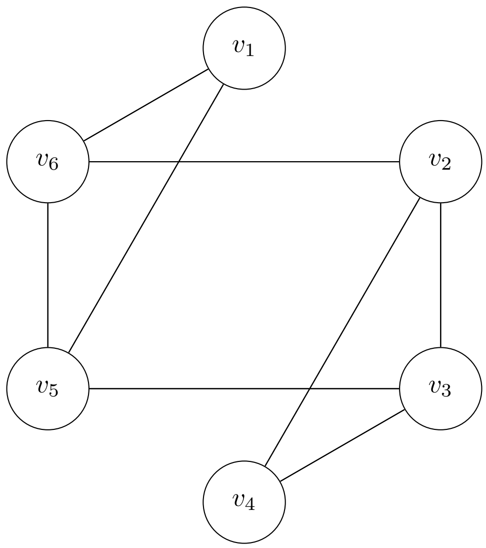
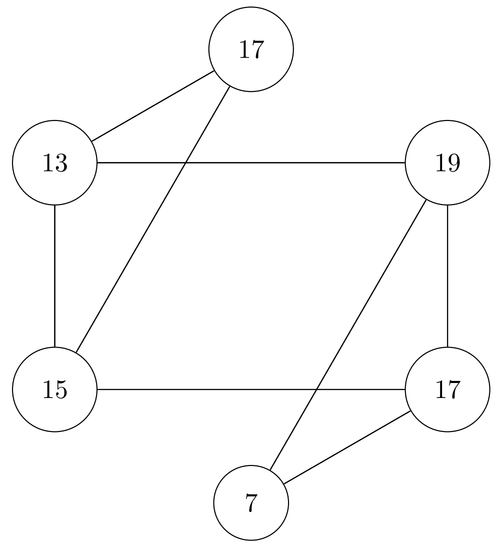
Computer Science Solutions Packet

1. Consider the following two vertex-weighted graphs, and denote them as having vertex sets $V = \{v_1, v_2, \dots, v_6\}$ and $W = \{w_1, w_2, \dots, w_6\}$, respectively (numbered in the same direction and way). The weights in the second graph are such that for all $1 \le i \le 6$, the weight of w_i is the sum of the weights of the neighbors of v_i. Determine the sum of the weights of the original graph.





Proposed by Misha Ivkov

Solution. This is the system of equations

$$\begin{aligned} v_5 + v_6 &= 17 && (1)\\ v_3 + v_4 + v_6 &= 19 && (2)\\ v_2 + v_4 + v_5 &= 17 && (3)\\ v_2 + v_3 &= 7 && (4)\\ v_1 + v_3 + v_6 &= 15 && (5)\\ v_1 + v_2 + v_5 &= 13 && (6) \end{aligned}$$

Adding either (2) and (6) or (3) and (5) gives $\boxed{32}$

2. Consider the natural implementation of computing Fibonacci numbers:

```
FUNCTION FIB(n):
    IF n = 0 OR n = 1 RETURN 1
    RETURN FIB(n − 1) + FIB(n − 2)
```

When FIB(10) is evaluated, how many recursive calls to FIB occur?

Proposed by Patrick Lin

Solution. Let $f(n)$ be the number of calls to `fib` during `fib`(n). Then

$$f(n) = 2 + f(n-1) + f(n-2)$$

with initial conditions $f(0) = 0$ and $f(1) = 0$. We can easily compute the recursion to get $f(10) = \boxed{176}$.

3. You are given the existence of an unsorted sequence $a_1, \dots, a_5$ of five distinct real numbers. The Erdos-Szekeres theorem states that there exists a subsequence of length 3 which is either strictly increasing or strictly decreasing. You do not have access to the a_i, but you do have an oracle which, when given two indexes $1 \le i < j \le 5$, will tell you whether $a_i < a_j$ or $a_i > a_j$. What is the minimum number of calls to the oracle needed in order to identify one such requested subsequence?

   Proposed by David Altizio

   Solution. We claim the answer is $\boxed{4}$. It is easy to see that three does not work; one can consider all possible sets of calls and for each one construct an ordering of the a_i which prevents determining a desired sequence. Now we exhibit a sequence of four calls which works. First call the oracle on (2,3), (3,4), and (2,4). This allows us to determine a total ordering of the numbers a_2, a_3, a_4. We now case on which one of these is the median. If it's a_3, (2,3,4) works. Otherwise, WLOG $a_3 < a_2 < a_4$. Now call (1,2). Then if $a_1 < a_2$, then (1,2,4) works; if $a_1 > a_2$, then (1,2,3) works. We are done.

4. Consider the grid of numbers shown below.

```
20 01 96 56 16
37 48 38 64 60
96 97 42 20 98
35 64 96 40 71
50 58 90 16 89
```

   Among all paths that start on the top row, move only left, right, and down, and end on the bottom row, what is the minimum sum of their entries?

   Proposed by Cody Johnson

   Solution. Notice that the path traversing down the fourth column has sum $\boxed{196}$, and it is not hard to see that there are no answers which are less than this.

   Remark. In general, such a problem can be solved pretty efficiently by a Dynamic Programming algorithm.

5. An *access pattern* π is a permutation of $\{1, 2, \dots, 50\}$ describing the order in which some n memory addresses are accessed. We define the *locality* of π to be how much the program jumps around the memory, or numerically,

$$\sum_{i=2}^{n} |\pi(i) - \pi(i-1)|.$$

   If π is a uniformly randomly chosen access pattern, what's the expected value of its locality?

   Proposed by Cody Johnson

   Solution. Let E denote the expected value of $|\pi(i) - \pi(i-1)|$. Then by linearity of expectation and symmetry, our answer is $(n-1)E$. Consider writing the numbers from 1 to n out. Then there are $n+1$ gaps between them. Since the places that two bars can be placed are independent and selected at random, the expected value of the size of the gap is $\frac{n+1}{3}$. Then the answer is $\frac{n^2-1}{3}$. Here the desired quantity is $\frac{50^2-1}{3} = \boxed{833}$.

6. We define $\mathcal{W}_{n,p}$ to be the complete weighted undirected random graph with vertex set $\{1, 2, \dots, n\}$: the edge (i,j) will have weight $\min(i,j)$ with probability p and weight $\max(i,j)$ otherwise. Let $\mathcal{L}_{n,p}$ denote the total weight of the minimum spanning tree of $\mathcal{W}_{n,p}$. Find the largest integer less than the expected value of $\mathcal{L}_{2018,1/2}$.

   Proposed by Misha Ivkov

   Solution. We prove that

$$\mathbb{E}[\mathcal{L}_{n,1/2}] = 2n - 3 + \frac{1}{2^{n-1}}$$

(where $\mathbb{E}$ denotes expected value) from which the answer follows.
Note that $\mathbb{E}[\mathcal{L}_{2,1/2}] = \frac{3}{2}$, which is true. Assume this statement is true for n. We show it follows for $n+1$. To do this, construct the MSP for the first n vertices. Adding an edge to the $n+1$st vertex will preserve the MSP structure, so $\mathbb{E}[\mathcal{L}_{n+1,1/2}] = \mathbb{E}[\mathcal{L}_{n,1/2}] + \mathbb{E}[s]$ where s is the weight of the edge to the $n+1$st vertex. This quantity is

$$\mathbb{E}[s] = \frac{n+1}{2^n} + \sum_{i=1}^{n} \frac{i}{2^i} = 2 - \frac{1}{2^n}$$

Then

$$\mathbb{E}[\mathcal{L}_{n+1,1/2}] = 2n - 1 + \frac{1}{2^n}$$

and we are done, so the answer is $2 * 2018 - 3 = \boxed{4033}$.

7. I give you a function **rand** that returns a number chosen uniformly at random from $[0, T]$ for some number T that you don't know. Your task is to approximate T. You do this by calling **rand** 100 times, recording the results as $X_1, X_2, ..., X_{100}$, and guessing

$$\hat{T} = \alpha \cdot \max\{X_1, X_2, ..., X_{100}\}$$

for some α. Which value of α ensures that $\mathbb{E}[\hat{T}] = T$?

Proposed by Cody Johnson

Solution. Lets calculate $\mathbb{E}[\hat{T}]$ when $\alpha = 1$. We have

$$\begin{aligned}
\mathbb{E}[\hat{T}] &= \int_0^T \Pr[\hat{T} > x]\,\mathrm{d}x \\
&= \int_0^T (1 - \Pr[\hat{T} \le x])\,\mathrm{d}x \\
&= \int_0^T (1 - \Pr[X_1 \le x \wedge X_2 \le x \wedge ... \wedge X_{100} \le x])\,\mathrm{d}x \\
&= \int_0^T (1 - (x/T)^{100})\,\mathrm{d}x \\
&= \frac{100}{101} T
\end{aligned}$$

Therefore, the answer is $\alpha = \boxed{\frac{101}{100}}$.

8. We consider a simple model for balanced parenthesis checking. Let $\mathcal{R} = \{\texttt{(())} \to \texttt{A}, \texttt{(A)} \to \texttt{A}, \texttt{AA} \to \texttt{A}\}$ be a set of rules for phrase reduction. Then the phrase is balanced if and only if the model is able to reduce the phrase to `A` by some arbitrary sequence of rule applications. For example, to show `((()))` is balanced we can do:

$$\texttt{((()))} \to \texttt{(A)} \to \texttt{A} \qquad \checkmark$$

Unfortunately, the above set of rules $\mathcal{R}$ is not complete; find the number of balanced parenthetical phrases of length 14 for which $\mathcal{R}$ **is insufficient** to show that they are balanced.

Proposed by Misha Ivkov and Patrick Lin

Solution. Let $f(n)$ be the number of phrases which can be shown to be balanced if the length is $2n$, with $f(1) = 0$ and let $g(n) = f(n)$, except $g(1) = 1$. Then we claim

$$f(n) = f(n-1) + \sum_{i=1}^{n-2} g(i) f(n-1-i)$$

This can be shown term by term. $f(n-1)$ represents taking all phrases of length $2n-2$ and adding a set of parens around them. For all the other terms, consider a phrase of length $2n$ as the combination of $(2k) \circ 2(n-k-1)$, with the first parentheses showing that it is indeed attainable. This is why $g(1) = 1$ is required, so that the first term exists. The number of ways to create $(2k) \circ 2(n-k-1)$ is $g(k)f(n-k-1)$, as suggested by the formula. Hence we can compute $f(7) = 37$, and the total number of balanced parenthetical phrases of length 14 is $\frac{1}{8}\binom{14}{7}$ so the answer is $\frac{1}{8}\binom{14}{7} - 37 = \boxed{392}$.

9. Consider the following modified algorithm for binary search, which we will call *weighted binary search*:

```
FUNCTION SEARCH(L, value)
    hi ← len(L) - 1
    lo ← 0
    WHILE hi ≥ lo
        guess ← ⌊w · lo + (1 − w) · hi⌋
        mid ← L[guess]
        IF mid > value
            hi ← guess - 1
        ELSE IF mid < value
            lo ← guess + 1
        ELSE
            RETURN guess
    RETURN -1 (not found)
```

Assume L is a list of the integers $\{1, 2, \ldots, 100\}$, in that order. Further assume that accessing the kth index of L costs $k+1$ tokens (e.g. $L[0]$ costs 1 token). Let S be the set of all $w \in [0.5, 1)$ which minimize the average cost when `value` is an integer selected at random in the range $[1, 50]$. Given that $S = \left(x, \frac{74}{99}\right]$, determine x.

Proposed by Misha Ivkov

Solution. Notice that all optimal values will have the property that $\lfloor wa + (1-w)b \rfloor = \lfloor \frac{74a+25b}{99} \rfloor$. This can be rewritten as

$$\lfloor w(a-b) \rfloor = \left\lfloor \frac{74}{99}(a-b) \right\rfloor$$

We know that not all (a, b) are possible as a result of running weighted binary search. Notice that $74^{-1} \equiv 95 \pmod{99}$, and $74 * (99 - 4n) \equiv n \pmod{99}$. This means that the largest $99 - 4n$ or a multiple thereof to appear as $b - a$ will give a lower bound on w (this value will dictate when the floor goes from one value to another). Consider the following steps:

$$(0, 99) \to (26, 99) \to (45, 99)$$

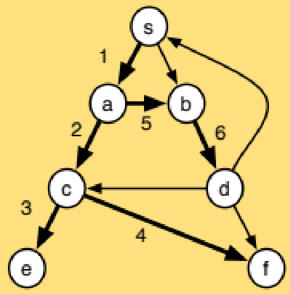
Here, $27 \mid b - a$. We can further check that no bigger values can appear. Hence to finish we just need to find y such that $w = \frac{y}{27}$. Notice that $\lfloor \frac{74}{99}(-54) \rfloor = -41$, so we want y such that $\lfloor -2y \rfloor = -40$, implying $y = 20$. Then the answer is $x = w = \boxed{\frac{20}{27}}$.

10. Consider a graph G with vertex set $\{v_1, v_2, \ldots, v_6\}$. Starting at the vertex v_1, an ant uses a DFS algorithm to traverse through G, under the condition that if there are multiple unvisited neighbors of some vertex, the ant chooses the v_i with smallest i. How many possible graphs G are there satisfying the following property: for each $1 \le i \le 6$, the vertex v_i is the i^{th} new vertex the ant traverses?

Proposed by David Altizio

Solution. To solve this problem, it is first important to recall what depth-first search (DFS) is. In a DFS algorithm, the ant will traverse through the vertices of a graph one at a time, travelling as far as possible before backtracking. This is done subject to the condition that the ant finishes searching a vertex if and only if all of that vertex's neighbors are already visited. The following example, taken from the **15-210 Parallel and Sequential Algorithms** course textbook, might serve as a good visual aid (although here the DFS is

Example 15.16. An example of DFS on a graph where the out-edges are ordered counterclockwise, starting from the left.



v	X
s	$\{\}$
a	$\{s\}$
c	$\{s, a\}$
e	$\{s, a, c\}$
f	$\{s, a, c, e\}$
b	$\{s, a, c, e, f\}$
d	$\{s, a, c, e, f, b\}$
c	$\{s, a, c, e, f, b, d\}$
f	$\{s, a, c, e, f, b, d\}$
s	$\{s, a, c, e, f, b, d\}$
b	$\{s, a, c, e, f, b, d\}$

Each row corresponds to one call to `DFS` in the order they are called, and v and X are the arguments to the call. In the last four rows the vertices have already been visited, so the call returns immediately without revisiting the vertices since they appear in X.

being done on an ordered graph as opposed to an unordered one). Here, X denotes the set of visited vertices ordered from left to right.

Note that the bolded part of the graph indicating the exact edges traversed in the DFS search forms a tree; this makes sense, since such an algorithm never visits a vertex twice (which in turn would create a cycle). The key to solving this problem is to focus on this underlying tree (called a *DFS tree*) and use this to develop a recursion that will help enumerate the number of graphs in question.

Consider any graph G on $n+1$ vertices $\{v_1, \ldots, v_{n+1}\}$ satisfying the property in the question. Delete the vertex v_{n+1} and all edges incident on it. Then the resulting graph G' on n vertices $\{v_1, \ldots, v_n\}$ also satisfies the property in question, since the DFS traversal for G must go through the vertices $v_1, \ldots, v_n$ first before finally reaching v_{n+1}. Now let P denote the unique path connecting vertex v_1 to vertex v_n in the DFS tree for G'. The crucial claim is that all of the neighbors of v_{n+1} must be entirely contained in P. Indeed, if this were not the case, then $\{v_j, v_{n+1}\} \in E(G)$ for some $v_j \notin P$. Note that by the property in the problem statement, in the DFS tree for G', v_j is traversed before v_n. But now this means that in the DFS tree for G, vertex v_{n+1} is traversed before v_n, since by definition the ant must visit all neighbors of v_j before backtracking onto P. This is a contradiction, and so indeed all neighbors of v_{n+1} must be located on the path P. In turn, one can construct a graph on $n+1$ vertices by first picking a vertex v in P for the ant to travel to v_{n+1} and then selecting any subset of the vertices in the path from v_1 to v not containing v in the DFS tree to add as extra neighbors to v_{n+1}. (These will not affect the correctness of the DFS tree due to the given ordering of the vertices, since the vertex v_{n+1} will still be traversed last.)

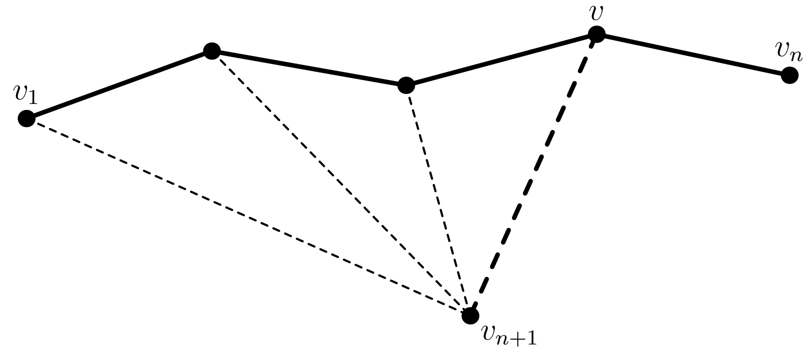
The second important step is to recongize that the number of choices for where to place the branching-off point is dependent on the length of P. Thus, for all positive integer pairs (n, k) with $1 \leq k \leq n-1$, let $G_{n,k}$ denote the number of connected graphs on $\{v_1, \ldots, v_n\}$ satisfying the following two properties:

- The DFS search starting from vertex v_1 traverses $v_1, v_2, \ldots, v_n$ in this order;
- In the DFS tree for the graph, the distance between vertices v_1 and v_n is k.

Additionally, define $G_{n,0} = 0$ for convienence purposes. Then in order to construct a graph counted in $G_{n+1,k+1}$, the vertex v_{n+1} must be attached to the vertex in the path from v_1 to v_n which is distance k away from v_1; this can only be done if the length of the path is at least k. Thus, by combining this with the logic above, one establishes the recursion

$$G_{n+1,k+1} = 2^k \sum_{j=k}^{n} G_{n,j}.$$

Figure 1: Adding the vertex v_{n+1} to the graph. Here the DFS tree is represented in bold. The remaining dashed edges are optional and lead to the 2^k term in the recurrence.



From here, it sufficees to compute the $G_{n,k}$ manually. The computation is a bit intensive when performing the calculations for the $n = 6$ case, but it is still doable.

$G_{n,k}$	0	1	2	3	4	5
1	0					
2	0	1				
3	0	1	2			
4	0	3	6	8		
5	0	17	34	56	64	
6	0	171	342	616	960	1024

The requested answer is the sum of the entries along the bottom row, which is $\boxed{3113}$.

Remark: The sequence of answers for various n - 1, 1, 3, 17, 171, 3113, 106419, ... - is sequence A015083 in the OEIS. No closed form is known, but it is known that the generating function $A(x)$ for this recurrence satisfies the equation

$$A(x) = \frac{1}{1 - xA(2x)}.$$

In this way, the sequence can be interpreted as a generalization of the Catalan number recurrence.