Задача 1.1:
проверяющие *Сергей Тарасов, Александр Кульков*
ответ правильный, но нет процедуры/объяснения того, что делать на k_0 вершинах явно – 0.5 балла.

Авторское решение: когда k_0 фиксировано и не является частью входа, задача может быть решена полным перебором за полиномиальное время. В вершинном покрытии и независимом множестве нужно перебирать все подмножества в V размера k_0 (которых $C_{|V|}^{k_0}$, а это полином степени k_0 от $|V|$), в простом пути нужно перебрать все наборы вершин размера k_0 и проверить, на них найдется простой путь.

Задача 1.2:
проверяющие *Даниил Селиханович, Павел Останин, Александр Кульков*
ответ без обоснования – 0 баллов,
правильный ответ и попытка привести пример – 0.5 балла,
правильный ответ и пример – 1 балл

Авторское решение: нет, в указанном классе есть и неразрешимые задачи.

Задача 1.3:
проверяющие *Даниил Селиханович, Александр Плавин*
не $\mathcal{NP}$ т.к. сертификат не полиномиален – 0 баллов,
$\mathcal{NP}$ т.к. сертификат полиномиален – 0 баллов,
замечено, что сертификат не успеет не считаться за время работы МТ – 0.2 балла.

Авторское решение: да, верно, т.к. из того, что верификатор $R(x, y)$ работает за полином от $|x|$, следует, что он не сможет прочитать сертификат целиком, если тот не ограничен некоторым фиксированным полиномом от $|x|$. Это означает, что указанный верификатор всегда читает кусок слова y, ограниченный некоторым полиномом от $|x|$, а значит, можно не учитывать остальную часть сертификата, а рассматривать только этот полиномиально ограниченный кусок слова за сертификат. Но тогда мы получаем, в точности определение класса $\mathcal{NP}$. Следовательно, определение $\widetilde{\mathcal{NP}}$ эквивалентно определению $\mathcal{NP}$.

Задача 1.4:
проверяющие *Сергей Тарасов, Михаил Гончаров*
определение сводимости – 0 баллов,
решение вида $w_1\#w_2\#\ldots\#w_n\# \in (L_1\#)^* \Leftrightarrow f(w_1)\#f(w_2)\#\ldots\#f(w_n)\# \in (L_2\#)^*$ без обоснования – 1 балл.

Авторское решение для первого варианта: да, верно. Пусть f — функция, сводящая L_1 к L_2. Тогда рассмотрим функцию g, которая отображает пустое слово в пустое; остальные слова, не заканчивающиеся на # в любое слово не из $(L_2\#)^*$, например в непустое слово, не содержащее на #;

$g : w_1\#\dots w_n\# \mapsto f(w_1)\#\dots f(w_n)\#$. Тогда слово из $(L_1\#)^*$ будет отображено в слово из $(L_2\#)^*$, а слово не из $(L_1\#)^*$ может или быть непустым и не оканчиваться на $\#$, тогда оно отображается не в $(L_2\#)^*$, или может иметь вид $w_1\#\dots w_n\#$, где хотя бы одно $w_i \notin L_1$ и тогда оно отображается не в L_2.

Задача 1.5:
проверяющий *Александр Плавин*
вариант $2 \to 3$:
«верно, т.к. 2-C$\in \mathcal{P}$» и тому подобные – 0 баллов,
неполная сводимость – 0.8 балла,
сводимость в обе стороны – 1 балл,
вариант $3 \to 2$:
неверно без указания мест либо за решение вида $\mathcal{NPC} \leqslant_p \mathcal{P}$,
сводимость не та, т.к. нельзя удалять вершины/литералы – 0.5 балла,
апелляция к $3-\mathsf{CNF}$ как к РОВНО $-\,3-\mathsf{CNF}$ – 0.5 балла,
обратная сводимость – 1 балл.

Авторское решение:
вариант $2 \to 3$: сводимость верная, вариант $3 \to 2$: – это сводимость в другую сторону.

Задача 1.6:
проверяющие *Даниил Селиханович, Павел Останин, Александр Кульков*
ответ без обоснования – 0 баллов,
правильный ответ и попытка привести пример – 0.5 балла,
правильный ответ и пример – 1 балл.

Авторское решение: Нет. Контрпримером служат $\log x$ и $\log 2x$.

Задача 2:
проверяющий *Александр Кульков*
правильное решение без обоснования монотонности/оценки/округления – 2 балла.

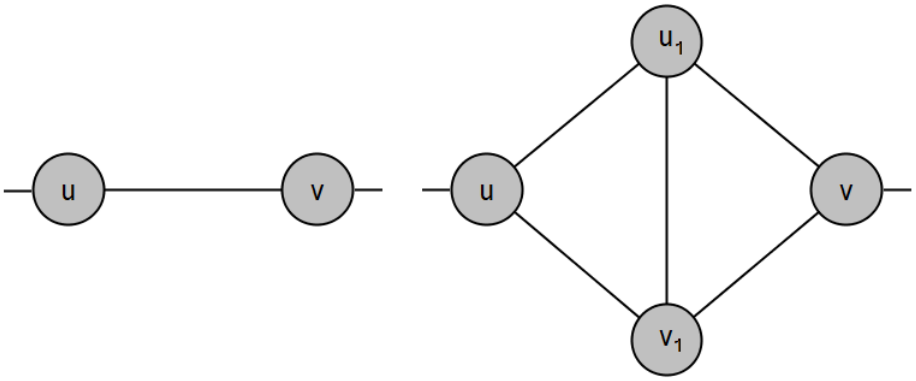
Авторское решение одного из вариантов: в данном случае асимптотика глубины дерева вызовов совпадает с асимптотикой $T(n)$. Заметим, что при отбрасывании $\log n$ из аргумента дерево становится длиннее, поэтому $T(n) \le \log_3 n$. Обратно, при замене $\frac{n}{3} - \log n$ на $\frac{n}{4}$ дерево становится короче, поэтому $T(n) \ge \log_4 n$. Итак, $T(n) = \Theta(\log n)$.

Отбрасывание округления обосновывается монотонностью. Монотонность следует по индукции из рассмотрения рекурренты $T(n) = aT(f(n)) + g(n)$ с монотонными f и g, причем $f(n) < n$. В нашем случае это выполняется.

Решение остальных вариантов аналогично.

Задача 3:
проверяющий *Сергей Шестаков*
принадлежность $\mathcal{NP}$ – +0.5 баллов,
корректная сводимость языка из $\mathcal{NPC}$ к данному – +3.5 баллов.

Авторское решение: Да, является. Принадлежность $\mathcal{NP}$ очевидна – сертификатом будут сами циклы. Построим полиномиальную сводимость $HC(1) \leq_P HC(4)$ (сводимость $HC \leq_P HC(8)$ строится аналогично). Сводящая функция по графу G строит граф $\tilde{G}$ следующим образом. Заменим все рёбра графа G по правилу: ребро (u, v) графа G преобразуется в «ромбик» (добавляются 2 вершины u_1 и v_1 вместе с рёбрами (u, u_1), (u, v_1), (u_1, v_1), (v_1, v), (u_1, v), а ребро (u, v) отбрасывается). Полученный граф обозначим $\tilde{G}$. Заметим, что если в исходном графе G существовал гамильтонов цикл, который содержит рёбра (x, y) и (u, v), то в графе $\tilde{G}$ мы можем каждый из «ромбиков» обойти одним из двух способов (например, «ромбик», сгенерированный из ребра (x, y), можно обойти ровно двумя способами: $x - x_1 - y_1 - y$ или $x - y_1 - x_1 - y$). В итоге получаем, что в графе $\tilde{G}$ существует хотя бы 2^n гамильтоновых циклов, т.е. он принадлежит $HC(4)$ и $HC(8)$ при $n \geq 3$. Обратно, пусть $\tilde{G}$ принадлежит, скажем, $HC(4)$. Тогда рассмотрим произвольный гамильтонов цикл в нём. Заметим, что «ромбик», соответствующий ребру (x, y) исходного графа, этот цикл может обойти одним из двух способов: $x - x_1 - y_1 - y$ или $x - y_1 - x_1 - y$. Следовательно, в исходном графе есть точно такой же гамильтонов цикл, но в котором «ромбики» заменены на соответствующие рёбра, то есть G принадлежит $HC(1)$.



Комментарии проверяющего: задача ожидаемо оказалось сложной, корректную сводимость написали около пятнадцати человек.

Я считал, что принадлежность $\mathcal{NP}$ должна быть указана явно, тем более, что много работ содержали некорректное рассуждения вида «сведём язык Гамильтонов путь/цикл к нашему языку, тем самым докажем, что наш язык $\mathcal{NP}$-полон». При этом в максимальные полбалла оценивались любые рассуждения или замечания о том, что язык принадлежит $\mathcal{NP}$ – фактически, про это нужно было просто вспомнить.

Теперь сводимость: правильные сводимости делились в основном на два типа.

1) Удвоить/размножить одну вершину (подобно конструкции в сводимости гамильтонова цикла к гамильтонову пути), затем на две размноженные вершины навесить что-то, у чего много гамильтоновых путей. Это что-то варьировалось по сложности начиная от клики фиксированного размера до нескольких копий исходного графа, соединённых специфическим образом. Идея следующая: если в графе есть гам. цикл, то есть путь из одной удвоенной вершины в другую. Этот путь замыкается навешиванием – причем благодаря наличию в навешенном графе многих путей у конечного графа получается много циклов. Наоборот, если после преобразования получилось много циклов, то доказывается, что цикл делится на два куска: кусок в навешенном графе и кусок в исходном графе с удвоенной вершиной. Последний же влечёт наличия цикла в исходном графе.

2) Решение, подобное авторскому: каждое ребро/каждая вершина исходного графа заменялись на некоторый граф (далее гаджет), причем разные рёбра/вершины заменялись на независимые гад-

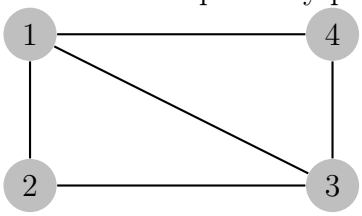
жеты. Тогда гамильтонов цикл в исходном графе генерировал множество циклов в преобразованном графе за счёт каждого такого гаджета. Обратная сводимость доказывалась благодаря независимости гаджетов – фактически, проход сквозь каждый гаджет в преобразованном графе эквивалентен проходу по ребру/через вершину в исходном графе.

Неправильные сводимости были достаточно разнообразными, однако несколько ошибок повторялись часто.

1) Рассуждение следующего вида: «сведём гамильтонов путь к нашему языку, пусть в исходном графе был гамильтонов цикл, тогда возьмём две смежные в этом цикле вершины ...» Это неверное рассуждение, никакая сводимость не может пользоваться конкретным циклом в гамильтоновом графе – просто потому, что непонятно как именно искать этот цикл. При построении можно пользоваться фактом того, что граф гамильтонов, но не самим гамильтоновым путём – он ведь не дан на вход.

Если подобное рассуждение непонятно, можно представить себе следующее: функция сводимости есть просто полиномиальный алгоритм, который вы должны написать. Как можно реализовать программно «возьмём две смежные в гамильтоновом цикле вершины» даже если вам дано, что где-то в графе гамильтонов цикл присутствует?

Если в качестве ребра бралось произвольное ребро графа, то такая сводимость также некорректна потому, что нет способа «заставить» некий существующий гамильтонов цикл в графе пройти по произвольно выбранному ребру. В качестве примера можно рассмотреть граф



Этот граф гамильтонов, однако никакой гамильтонов цикл (он всего один) не проходит по ребру $1 - 3$.

Совершивших подобную ошибку и одновременно желающих разобраться я призываю решить (аккуратно решить!) две задачи:

– свести Клику к Клике-на-половине-вершин (язык графов, в которых есть клика ровно на половине вершин графа),

– свести гамильтонов путь/цикл к двойному гамильтонову маршруту (язык графов, в которых есть замкнутый маршрут – он может дважды проходить по одному ребру, – проходящий через все вершины графа ровно по два раза).

2) Некорректная сводимость – сводимость, верная в одну сторону (чаще всего тривиально верная), но неверная в другую. Эта ошибка была частой, при этом сводимости (и даже то, в какую сторону сводимость верна) – крайне разнообразными. Подобное решение оценивалось в 0 баллов по следующей логике: в частичный балл оценивается решение неполное – то, которое додумыванием, дописыванием или подобными усилиями можно привести к правильному. Некорректная сводимость не может быть «исправлена» до корректной никак иначе, кроме как полным переписыванием.

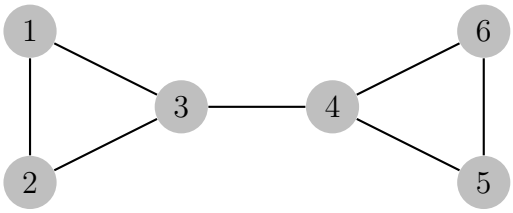
Неполный балл из 3.5 за сводимость можно было получить лишь за корректную сводимость, обоснование которой неполно/недописано/неясно – это удалось паре человек.

Типичная ошибка, приводящая к некорректной сводимости, имела вид «пусть в исходном графе не было гамильтонова цикла, тогда в преобразованном цикла и не появится/появится не более одного цикла» – это утверждение не обосновывается и (скорее всего) не является верным.

Типичными контрпримерами, часто помогающими понять суть проблемы, являются:

– граф-путь на n вершинах – можно на трёх или четырёх,

– граф «песочные часы» – он имеет четыре различных гамильтоновых пути, но не имеет ни одного гамильтонова цикла



Задача 4.1:

проверяющий *Павел Останин*

выписана система с двумя уравнениями и двумя неизвестными – +1 балл,

представление a_n и b_n очевидными суммами из представления $(1+\sqrt{d})^n$ биномом Ньютона не оценивалось (такую сумму необходимо свернуть),

По техническим причинам в этом пункте задачи результат, выставленный на работе, умножается на 1.5 – т.е. оценивается из максимума в 3 балла

Задача 4.2:

проверяющий *Павел Останин*

решение рекурренты – +1 балл,

асимптотика – +1 балл

в некоторых работах выписывались a_n и b_n очевидными суммами из представления $(1+\sqrt{d})^n$ биномом Ньютона, а затем констатировалось совпадение асимптотик ($a_n = \Theta(b_n)$); такой подход оценивался из 1 балла,

несколько студентов после получения системы из двух уравнений с двумя неизвестными последовательностями сразу переходили к производящим функциям, и из них находили a_n и b_n и их асимптотики; в этом случае второй пункт оказывался формально выполненным, а первый — лишь частично: в нём явно требовалось выписать по одному уравнению на каждую из переменных, что проделано не было (хотя это легко можно сделать, найдя явный вид a_n и b_n в виде сумм экспонент: известны оба корня характеристического уравнения). Такой подход оценивался из половины баллов за $4(i)$ и полных баллов за $4(ii)$.

Авторское решение:

(i) С одной стороны, $(1+\sqrt{3})^{n+1} = a_{n+1} + \sqrt{3}b_{n+1}$, а с другой можем использовать сведения о коэффициентах в случае степени на 1 меньше и получить $(1+\sqrt{3})^n(1+\sqrt{3}) = (a_n+\sqrt{3}b_n)(1+\sqrt{3}) = (a_n+3b_n)+\sqrt{3}(a_n+b_n)$. Отсюда имеем систему вида: $\begin{cases} a_{n+1} = a_n + 3b_n; \\ b_{n+1} = a_n + b_n. \end{cases}$

Заметим теперь, что из первого уравнения $b_n = \dfrac{1}{3}(a_{n+1} - a_n)$, а тогда подстановка во второе уравнение даст одно уравнение только для a_n: $\dfrac{1}{3}(a_{n+2} - a_{n+1}) = a_n + \dfrac{1}{3}(a_{n+1} - a_n)$, откуда $a_{n+2} - 2a_{n+1} - 2a_n = 0$.

Аналогично, из второго уравнения $a_n = b_{n+1} - b_n$, а значит, $(b_{n+2} - b_{n+1}) = (b_{n+1} - b_n) + 3b_n$, откуда $b_{n+2} - 2b_{n+1} - 2b_n = 0$. Рекуррентные уравнения для последовательностей (без учёта начальных условий) совпали.

(ii) Полученные рекуррентные уравнения можно решить явно, вычислив по малым n соответствующие начальные условия. Пойдем другим путём. Заметим, что $(1-\sqrt{3})^n = a_n - \sqrt{3}b_n$, а поэтому достаточно сложить это равенство с исходным для получения $a_n = \dfrac{1}{2}(1-\sqrt{3})^n + \dfrac{1}{2}(1+\sqrt{3})^n$. Аналогично, вычитанием из первого уравнения второго найдем $b_n = \dfrac{1}{2\sqrt{3}}(1+\sqrt{3})^n - \dfrac{1}{2\sqrt{3}}(1-\sqrt{3})^n$.

Ясно, что $a_n = \Theta(b_n)$.

Решение второго варианта аналогично первому: (i) $\begin{cases} a_{n+1} = a_n + 5b_n; \\ b_{n+1} = a_n + b_n. \end{cases}$

Итого $a_{n+2} - 2a_{n+1} - 4a_n = 0$, $b_{n+2} - 2b_{n+1} - 4b_n = 0$.

(ii) $a_n = \frac{1}{2}(1-\sqrt{5})^n + \frac{1}{2}(1+\sqrt{5})^n$, $b_n = \frac{1}{2\sqrt{5}}(1+\sqrt{5})^n - \frac{1}{2\sqrt{5}}(1-\sqrt{5})^n$.

Как и в первом варианте, $a_n = \Theta(b_n)$.

Задача 5.1:

проверяющие *Александр Иванов, Александр Кульков*

идеи вида «найдём первую единицу, применим любой шаблон, повторим» – 0 баллов,

сформулирована идея обнулять по одному биту и сделана попытка получить подобное как комбинацию конкретных шаблонов – до 1 балла,

показано, как получить одну единицу – 2 балла.

Задача 5.2:

проверяющие *Александр Иванов, Александр Кульков*

правильный ответ – +1 балл,

обоснование ответа – +1 балл.

Задача 5.3:

проверяющие *Александр Иванов, Александр Кульков*

сведение к СЛАУ без уточнений какую именно систему решаем или без надлежащего обоснования корректности – 1-2 балла.

Авторское решение.

1. Научимся получать $b = 1$. Если мы можем получить его, то можем получить любой другой шаблон.

   Один из вариантов:

   (a) Применим шаблон 1 0000 1 0000 1 0000 1 0000 1 к позиции 1

   (b) Применим шаблоны 11111 в следующем порядке:

      i. 1[0000 1]0000 1 0000 1 0000 1 → 1[1111 0]0000 1 0000 1 0000 1
      ii. 1 1111[0 0000]1 0000 1 0000 1 → 1 1111[1 1111]1 0000 1 0000 1
      iii. 1 1111 1 1111 1[0000 1]0000 1 → 1 1111 1 1111 1[1111 0]0000 1
      iv. 1 1111 1 1111 1 1111[0 0000]1 → 1 1111 1 1111 1 1111[1 1111]1

   (c) В итоге получим на ленте 21 единиц подряд. Ещё 4 раза приложим шаблон 11111, чтобы занулить все, кроме первой.

2. Сдвигу шаблона на k соответствует умножение на x^k, а композиции шаблонов соответствует сумме многочленов:
$$A(x) = x^{j_1}A_{i_1}(x) + x^{j_2}A_{i_2}(x) + \cdots + x^{j_n}A_{i_n}(x)$$

3. Согласно прошлому пункту если первый шаблон представлен многочленом $A_1(x)$, а применяли мы его в позициях $j_1, \ldots, j_n$, то строка, полученная на ленте будет соответствовать следующему многочлену:
$$(x^{j_1} + \cdots + x^{j_n})A_1(x) = P_1(x)A_1(x)$$

Соответственно, если мы используем различные шаблоны, их композиция может быть представлена в виде:

$$P_1(x)A_1(x) + \cdots + P_m(x)A_m(x) = B(x)$$

Здесь $P_k(x)$ — «позиционный» многочлен шаблона k, который состоит из слагаемых вида x^j, где j — позиция приложения шаблона a_k. Как и в случае с числами, такое уравнение будет разрешимо тогда и только тогда, когда многочлен-образец $B(x)$ делится на наибольший общий делитель многочленов-шаблонов $A_k(x)$. Обоснование: Можно поочерёдно выводить из рассмотрения пары шаблонов $A_1(x)$ и $A_2(x)$, заменяя их шаблоном $G(x) = \gcd(A_1, A_2)$, так как с одной стороны он может быть получен как линейная комбинация A_1 и A_2 алгоритмом Евклида, а с другой любой многочлен вида $B_1(x)A_1(x) + B_2(x)A_2(x)$ делится на $G(x)$ и может быть представлен в виде $B_3(x)G(x)$.

Задача 6.1:
проверяющий *Эдуард Горбунов*
оценены вероятности того, что будет сделано $\geqslant k$ итераций, либо показано, что такая ситуация возможна – 1 балл

Задача 6.2:
проверяющий *Эдуард Горбунов*
ситуация, когда строго не объяснено, что все перестановки равновероятны – 1 балл,
показано, что получается случайная перестановка с вероятностью $1/n!$ – 2 балла

Задача 6.3:
проверяющий *Эдуард Горбунов*
получена рекуррента на мат. ожидание – 1 балл,
правильные рассуждения, но технические детали пропущены – 3 балла

Авторское решение:

(i) Для любого $k > 0$ есть ненулевая вероятность, что алгоритм проработает хотя бы k шагов. Следовательно, искомый супремум равен ∞.

(ii) Результатом работы процедуры является массив $C[1..n]$, который является случайной перестановкой массива $A[1..n]$. Нужно доказать, что вероятность каждой из перестановок исходного массива равна $\frac{1}{n!}$. Зафиксируем некоторую перестановку $A[\sigma(1)], A[\sigma(2)], \ldots, A[\sigma(n)]$ (здесь $\sigma(i)$ — индекс i-го массива A в указанной перестановке). Когда $i = 1$, нам подходит только один вариант, при котором в строчке 6 генерируется $j = \sigma(1)$. Вероятность этого события (обозначим его B_1) равна $\frac{1}{n}$. Далее счётчик i увеличивается на единицу. Рассмотрим первый момент времени, когда выполняется строчка 6 и $i = k > 1$, считая, что первые $k-1$ элемент массива C соответствуют выбранной перестановке массива A. Нам подходят только такие исходы: в строчке 6 случайный генератор возвращает l раз ($l = 0, 1, 2, \ldots$) подряд индекс из $\{\sigma(1), \ldots, \sigma(k-1)\}$ (обозначим это событие через $A_{k,l}$; отметим, что при этом индекс i не будет меняться), а в $(l+1)$-й раз возвращает $\sigma(k)$ (событие B_k). Вероятность такого события (обозначим его B_k) равна $\sum_{l=0}^{\infty} \underbrace{\left(\frac{k-1}{n}\right)^l}_{\text{вер-ть } A_{k,l}} \underbrace{\frac{1}{n}}_{\text{вер-ть } B_k} = \frac{1}{1-\frac{k-1}{n}}\frac{1}{n} = \frac{1}{n-k+1}$. Заметим, что вероятность события B_k имеет ровно такой же вид и в случае $k = 1$. Получаем, что вероятность нужной нам перестановки

равна произведению вероятностей событий B_k для $k = 1, 2, \ldots, n$, т.е. $\frac{1}{n!}$, что и требовалось доказать.

(iii) Рассмотрим случайные величины: T — число вызовов функции RAND в строчке 6 и T_k — число вызовов функции RAND в строчке 6 при $i = k$. Заметим, что $T = \sum_{k=1}^{n} T_k$. Пользуясь линейностью математического ожидания, получаем $\mathbf{E}[T] = \mathbf{E}\left[\sum_{k=1}^{n} T_k\right] = \sum_{k=1}^{n} \mathbf{E}[T_k]$. Найдём $\mathbf{E}[T_k]$. Заметим, что $T_1 \equiv 1$, а значит, $\mathbf{E}[T_1] = 1$. Пусть $k > 1$, $A'_{k,l}$ — событие, при котором l раз подряд при исполнении строчки 6 при $i = k$ сгенерировались индексы j, для которых $B[j] =$ TRUE, а в $(l+1)$-й раз сгенерировался j, для которого $B[j] =$ FALSE. Тогда получаем

$$\mathbf{E}[T_k] = \sum_{l=0}^{\infty} (l+1) \underbrace{\left(\frac{k-1}{n}\right)^{l} \left(1 - \frac{k-1}{n}\right)}_{\text{вер-ть } A'_{k,l}} = \frac{1}{\left(1 - \frac{k-1}{n}\right)^2} \left(1 - \frac{k-1}{n}\right) = \frac{1}{1 - \frac{k-1}{n}} = \frac{n}{n-k+1},$$

что согласуется с полученным ранее $\mathbf{E}[T_1] = 1$. Следовательно,

$$\mathbf{E}[T] = \sum_{k=1}^{n} \frac{n}{n-k+1} = n\left(\ln n + \gamma + O\left(\frac{1}{n}\right)\right) = n \ln n + \gamma n + O(1) = \Theta(n \log n),$$