1. Известны открытые ключи Алисы $(107, 187)$ и Боба $(7, 253)$. Алиса хочет послать сообщение 17 Бобу и подписать его своей подписью. Вычислите зашифрованное сообщение Алисы и его цифровую подпись.

$Cm = P_B(17) = 17^7(mod253) = 17 \cdot (17^3)^2(mod253) = 17 \cdot (17 \cdot 17^2)^2(mod253) \equiv 17 \cdot (17 \cdot 36)^2(mod253) \equiv 17 \cdot (106)^2(mod253) \equiv 17 \cdot 104(mod253) \equiv 250(mod253)$ – то есть 250 – это зашифрованное сообщение Алисы Бобу.

$\phi_A(N) = 10 \cdot 16 = 160$

$e = d^{-1}mod160 \Rightarrow ed = 1mod160 \Rightarrow 107d + 160n = 1$. Методом пристального взгляда спокойно находится решение: $d = 3, n = -2$.

Тогда секретный ключ Алисы $S_A = (3, 187)$. С помощью него мы можем с легкостью узнать цифровую подпись!

$s = 17^3mod187 \equiv 17 \cdot 289mod187 \equiv 17 \cdot 102mod187 \equiv 51mod187 = 51$.

Таким образом, мы получаем цифровую подпись сообщения, равную 51.

2. Вы хотите, чтоб некто M подписал своей электронной подписью сообщение x. Однако, очевидно, вы не добьётесь результата, послав M сообщение x, поскольку оно выглядит подозрительно. Однако, пусть (e, n) открытый ключ M, а (d, n) — его секретный ключ (d вам неизвестно).

Возьмём случайное число r по модулю n и составим сообщение $y = r^e x \pmod n$. Предположим, что y выглядит достаточно невинно, для того, чтобы M согласился подписать y своей электронной подписью и переслать вам подписанную версию: s_y. Если M подпишет сообщение, то как по подписанному сообщению s_y и известным вам данным получить правильную подпись для сообщения x?

$y = r^e x(modn) = (r \cdot x^{\frac{1}{e}})^e modn$

M посылает электронную подпись $s_y = y^d modn$, откуда $s_y = nk + y^d \Rightarrow n = \frac{s_y - y^d}{k}$

Мы знаем, что $ed = 1mod\phi(n)$. Отсюда $d = \frac{1}{e}modn$.

Правильная подпись для сообщения x: $s_x = x^d modn = x^{\frac{1}{e}modn}modn$ (n мы можем найти из формулы выше).

3. Алиса и три её друга используют криптосистему RSA. При этом её друзья используют открытые ключи $(N_i, 3)$ с возведением в степень 3, и $N_i = p_i q_i$ для случайно выбранных n-битовых простых чисел p_i и q_i. Покажите, что если Алиса пошлёт одно и то же n-битовое сообщение M всем троим, то перехватившая все три закодированных сообщения Ева (и знающая открытые ключи) сможет быстро (полиномиально) восстановить M.
Указания. Пусть модули попарно взаимнопросты. Как, зная зашифрованные сообщения, получить значение $M^3 \pmod{N_1 N_2 N_3}$?

Когда **Ева** перехватит сообщения, у нее будут данные $M^3(modN_1)$, $M^3(modN_2)$ и $M^3(modN_3)$.

Далее **Ева** пользуется Китайской теоремой об остатках, с помощью которой узнает значение $M^3 mod(N_1 \cdot N_2 N_3)$.

После этого **злой Еве** остается лишь извлечь кубический корень из полученного результата M^3, и таким образом она восстановит M.

4. Ева решила подобрать секретный ключ Алисы (d, N) с помощью вероятности: она выбирает случайное число от 2 до $N - 1$ и проверяет, подходит ли оно на роль d за полиномиальное время. Оцените ассимптотически математическое ожидание числа попыток Евы. Является ли её алгоритм более эффективным (в среднем), чем полный перебор?

В алгоритме Евы события, что Ева подобрала нужное числа, независимы, поэтому мат ожидание суммы равно сумме мат ожиданий, то есть $\sum_1^k \frac{1}{N-2}$, где k – среднее количество попыток, после которых Ева найдет ключ. Нетрудно догадаться, что $k = N-2$.

В случае с простым перебором вероятность достать ключ на i-той попытке равна $\frac{1}{N-1-i}$ – видно, что это даже эффективнее алгоритма Евы (хотя по ассимптотике они равны).

Значит, алгоритм Евы не является в среднем более эффективным, чем полный перебор.

5. Докажите, что алгоритм, заданный псевдокодом строит случайное m-элементное подмножество множества $\{1, \ldots, n\}$. То есть, что `RandomSample(m, n)` равновероятно возвращает каждое m-элементное подмножество, в предположении, что `Random(1,n)` случайная величина, возвращающая с равной вероятностью числа от 1 до n.

```
Function RandomSample(m, n) :
    if m == 0 then
        return ∅
    else
        S = RandomSample(m − 1, n − 1);
        i = Random(1, n);
        if i ∈ S then
            return S ∪ {n}
        else
            return S ∪ {i}
        end
    end
end
```

Дисклеймер: можно сразу перемотать решение на конец, т.к. там я расписал решение, которое мне пришло в голову непосредственно перед отправкой. Но если у читателя есть желание развлечься, то при желании можно и прочитать ту тяжелую дичь, которую я написал изначально.

jmp метка

Сразу отметим, что этот алгоритм можно изобразить с помощью рекурсивного дерева из одной ветки (делать мы это, конечно же, не будем).

Заметим, что дерево у нас растет снизу вверх (логично): от листьев к корню (что?).

Когда мы окажемся в самом низу, значения наших m и n будут следующими: 1 и $n-m+1$. То есть i в этой вершине может с равной вероятностью принимать значения от 1 до $n-m+1$. И мы добавляем его в наше множество.

Далее на второй вершине i принимает с равной вероятностью значения от 1 до $n-m+2$, но немного не совсем. Если выпадает число, которое уже в множестве, то мы добавляем в множество $n-m+2$. Таким образом, у этого числа вероятность выпасть в два раза больше, чем у остальных.

И так далее до корня дерева.

Таким образом, у нас получается вероятность для $n-m+1$ чисел: $\frac{1}{n-m+1}, \frac{1}{n-m+2}, \frac{1}{n-m+3}, \ldots, \frac{1}{n}$.

Вероятность для $n-m+i$-ого числа: (i - 1) нулей, $\frac{i}{n-m+i}, \frac{1}{n-m+i+1}, \ldots \frac{1}{n}$.

И соответственно для m-того числа: $\frac{m}{n}$.

Математическое ожидание каждого элемента будет вычисляться с помощью суммы $A_i + \overline{A_i} \cdot B_i + \overline{A_i B_i} \cdot C_i + \cdots + \overline{A_i B_i} \ldots \overline{Y_i} \cdot Z_i$.

Несложно заметить, что $A_i + \overline{A_i} \cdot B_i = B_i(A_i + \overline{A_i}) + (1 - B_i) \cdot A_i = B_i + A_i + 0 = A_i + B_i$ – вероятности i-ого и $i+1$-ого равны после i-ого шага.

И аналогично можно расписать вероятность попадания в множество каждого числа. У нас получится $\frac{m}{n}$ для каждого числа, откуда следует, что все m-элементные подмножества равновероятны.

метка:

Итак, для i-ого числа найдем вероятность, что оно не попало в подпоследовательность:

$\frac{n-1}{n} \cdot \frac{n-2}{n-1} \cdot \cdots \cdot \frac{n-m+i}{n-m+i+1} \cdot \frac{(n-m+i)-i}{n-m+i}$.

Ну и тут все числители и знаменатели (кроме первого знаменателя и последнего числителя) сокращаются, и мы получаем, что вероятность НЕ попасть в множество i-того члена равна $\frac{n-m}{n}$ – не зависит от i! (это восклицательный знак).

То все элементы с равной вероятностью НЕ попадут в подмножество, откуда следует, что они с равной вероятностью попадут в подмножество (вроде очевидно, но на всякий случай проверим).

Каждый элемент попадет в подмножество с вероятностью $1 - \frac{n-m}{n} = \frac{m}{n}$ – что вполне логично, т.е. мат ожидание количества элементов в подмножестве в таком случае ровно m.

6. Рандомизированный алгоритм поиска k-й порядковой статистики на каждом шаге делает partition по случайному элементу отрезка массива (если в нём более одного элемента) и рекурсивно вызывается либо для левого, либо для правого отрезка получившегося разбиения. Докажите, что математическое ожидание времени работы алгоритма есть $O(n)$, используя анализ индикаторных случайных величин $X_{i,j,k}$, возвращающих 1, если i-я порядковая статистика массива сравнивалась с j-й (при поиске k-й порядковой статистики).

Указания.

1. Получите явную формулу для $\mathrm{E}[X_{i,j,k}]$.

2. Пусть X_k — случайная величина, возвращающая число всех сравнений при поиске k-й порядковой статистики. Покажите, что

$$\mathrm{E}\left[X_k\right] \leqslant 2\left(\sum_{i=1}^{k}\sum_{j=k}^{n}\frac{1}{j-i+1} + \sum_{j=k+1}^{n}\frac{j-k-1}{j-k+1} + \sum_{i=1}^{k-2}\frac{k-i-1}{k-i+1}\right)$$

3. Докажите $\mathrm{E}[X_{i,j,k}] \leqslant 4n$.