Вказівки щодо розв'язання завдання № 30 відбірково-тренувальних зборів команди міста Києва

1.Іншопланетний словник

Усі підходи, де потрібно генерувати всі можливі слова, працюватимуть занадто повільно, щоб розв'язати задачу за вказаний в умові час.

Позначимо через:

$f(x)$ — кількість слів, «важливість» яких строго більша за x;

$g(x, s)$ — кількість слів, «важливість» яких дорівнює x і які починаються на рядок тексту s.

Використовуючи двійковий пошук і функцію f можна знайти яку важливість має слово на K-ому місці в словнику. Далі можна відновити саме K-те слово за допомогою функції g. Псевдокод має такий вигляд:

```
 1. A := (мінімально можлива важливість) - 1
 2. B := (максимально можлива важливість)
 3. поки B - A > 1
 4.     C := (A + B) div 2
 5.     якщо f(C) < K
 6.         тоді:   B := C
 7.         інакше: A := C
 8. L  := K - f(B)
 9. L1 := 0, s := ""
10. поки довжина(s) < M
11.     послідовно для всіх літер a
12.         якщо L1 + g(B, s + a) < L
13.             тоді:   L1 := L1 + g(B, s + a)
14.             інакше: s := s + a
15.                     вийти з циклу за літерами
16. повернути s
```

Маємо:

- на кроці 1 $f(A) = N^M$;
- на кроці 2 $f(B) = 0$;
- впродовж роботи алгоритму справджуються такі нерівності:
  - $f(A) \geq K$;
  - $f(B) < K$;
  - $A < B$, як наслідок попередніх двох нерівностей;
- крок 3 — умова завершення двійкового пошуку, коли $B = A + 1$, $f(A) \geq K$, $f(B) = f(A + 1) < K$. Це означає, що саме K-те слово в словнику прибульців має важливість B;
- кроки 4–7 — тіло алгоритму двійкового пошуку;
- на кроці 8 ми обчислюємо, який номер має шукане слово у списку лексикографічно відсортованих слів, кожне з яких має важливість B;
- на кроці 9 ми задаємо початкові умови для пошуку шуканого слова. L_1 — кількість слів з важливістю B (такою самою, як і в шуканого слова), що розташовані при алфавітному впорядкуванні раніше за s;
- на кроці 10 ми задаємо умови виходу: тільки s матиме довжину M, тобто буде повністю визначено словом з іншопланетного словника, наш алгоритм закінчить свою роботу;

- на кроці 11 ми послідовно перебираємо усі літери іншопланетного алфавіту для того, щоб визначити літеру, яка буде наступною в *s*;
- на кроках 12–15 ми аналізуємо, скільки слів починається на (*s* + *a*). Якщо разом з усіма словами, які йдуть раніше за (*s* + *a*) їх буде менше за *L*, то шукане слово починається на *s* + *b*, де *b* в алфавітному порядку розташовано після *a*. Інакше *a* — наступна літера слова.

Опишемо побудову функцій *f* і *g*, ґрунтуючись на можливості побудови частин слів — префіксів і суфіксів. Знаючи всі можливі суфікси довжини *S*, можна обчислити всі їхні важливості, впорядкувати за спаданням «важливості» і двійковим пошуком в масиві довжиною N^S визначати, скільки існує суфіксів довжини *S* з «важливістю»:

- або строго більшою за *x* — для знаходження *f*;
- або однаковою з *x* — для знаходження *g*.

Для розв'язання визначимо *S* як результат округлення *M*:2 до найближчого цілого числа. У цьому випадку $S \le 5$, $M - S \le 5$, а кількості префіксів і суфіксів не перевищують $16^5 = 1048576$.

Попередня підготовка виглядає так:

```
1. suf_sums := порожній список
2. для кожного можливого слова довжини S, позначеного як str
3.     sum := важливість str, якщо воно знаходиться в кінці слова
4.     додати в suf_sums обчислену sum
5. впорядкувати за неспаданням suf_sums
```

Скористаємося допоміжною функцією *h*(*x*) — кількість чисел у suf_sums, які строго менші за *x*. Для впорядкованого масиву suf_sums програмування *h*(*x*) зводиться до двійкового пошуку. При використанні C++ можна реалізувати цю функцію таким рядком:

```
return lower_bound(suf_sums.begin(), suf_sums.end(), x) - suf_sums.begin().
```

Тоді функцію *f*(*x*) знаходять таким чином:

```
1. count := 0
2. для кожного можливого слова довжини M - S, позначеного як str
3.   sum   := важливість str, якщо воно знаходиться на початку слова
4.   count := count + (suf_sums.size() - h(x - sum + 1))
5. повернути count
```

Тут на кроці 4 додається кількість суфіксів, важливість яких строго більша за *x* – *sum*.

Функцію *g*(*x*, *str*) знаходять таким чином:

```
 1. count := 0;
 2. якщо довжина(str) <= M - S
 3. тоді:
 4.для кожного можливого слова довжини M - S - довжина(str), позначеного як str2
 5.      sum := важливість (str + str2), якщо воно знаходиться на початку слова
 6.    count := count + h(x - sum + 1) - h(x - sum)
 7. інакше:
 8. для кожного можливого слова довжини M - довжина(str), позначеного як str2
 9.         sum := важливість (str + str2)
10.         якщо sum = x
11.         тоді: count := count + 1
12. повернути count
```

2. Симетричні візерунки

Розглянемо послідовність відстаней між сусідніми точками у порядку зростання координати x. Потрібно з'ясувати, чи можна видалити з цієї послідовності одне число таким чином, щоб по обидві сторони від видаленого числа частини послідовності були симетричними. Видалення крайнього числа відповідає випадку, коли один з візерунків складається лише з однієї точки. Але зовсім не видаляти число не можна, навіть якщо початкова послідовність симетрична, бо згідно з умовою кожен візерунок має бути непорожнім. Нехай $d_1\, d_2 \ldots d_{n-1}$ — отримана послідовність відстаней. Достатньо з'ясувати:

- при яких $i = 1, 2, \ldots, n-1$ рядок $d_1\, d_2 \ldots d_i$ є симетричним;
- при яких $j = 1, 2, \ldots, n-1$ рядок $d_j\, d_{j+1} \ldots d_{n-1}$ є симетричним.

Знаючи це, легко знайти місце можливого поділу рядка з найменшою координатою x або зробити висновок, що такого місця немає взагалі.

Пряма перевірка всіх послідовностей $d_1\, d_2 \ldots d_i$ та $d_j\, d_{j+1} \ldots d_{n-1}$ потребує $O(n^2)$ операцій, що неприйнятно при n порядку 10^5. Але існує спосіб здійснити таку перевірку набагато швидше. Опишемо його лише для послідовностей вигляду $d_1\, d_2 \ldots d_i$, бо послідовності виду $d_j\, d_{j+1} \ldots d_{n-1}$ перевіряють аналогічно. Симетричними є ті й лише ті рядки $d_1\, d_2 \ldots d_i$, які є кінцями рядка $d_{n-1}\, d_{n-2} \ldots d_1$. Пошук усіх таких рядків є незначною модифікацією алгоритму Кнута — Моріса — Пратта пошуку входжень одного рядка в іншому.

Для кожної послідовності виду $d_1\, d_2 \ldots d_i$ визначимо її *префікс-функцію*: найбільше натуральне $k < i$, при якому послідовності $d_1\, d_2 \ldots d_k$ і $d_{i-k+1}\, d_{i-k+2} \ldots d_i$ збігаються. Якщо таких натуральних k не існує, то префікс-функція дорівнює 0.

Позначимо префікс-функцію рядка $d_1\, d_2 \ldots d_i$ через $\mathrm{Prf}(i)$. Покажемо, як швидко обчислити $\mathrm{Prf}(i)$ для всіх i від 1 до $n-1$. Покладемо $\mathrm{Prf}(0) = -1$. Це буде зручним при використанні подальших формул. Послідовно обчислимо $\mathrm{Prf}(i)$ для $i = 1, 2, \ldots, n-1$. Для кожного натурального $i > 1$ з рівності послідовностей

$$d_1\, d_2 \ldots d_{\mathrm{Prf}(i)} = d_{i-\mathrm{Prf}(i)+1}\, d_{i-\mathrm{Prf}(i)+2} \ldots d_i$$

випливає рівність їхніх підпослідовностей:

$$d_1\, d_2 \ldots d_{\mathrm{Prf}(i)-1} = d_{i-\mathrm{Prf}(i)+1}\, d_{i-\mathrm{Prf}(i)+2} \ldots d_{i-1}.$$

Звідси маємо: $\mathrm{Prf}(i-1) \geq \mathrm{Prf}(i) - 1$, або ж $\mathrm{Prf}(i) \leq \mathrm{Prf}(i-1) + 1$.

Рівність $\mathrm{Prf}(i) = \mathrm{Prf}(i-1) + 1$ справджується тоді й лише тоді, коли $d_{\mathrm{Prf}(i-1)+1} = d_i$.

Інакше, тобто при $d_{\mathrm{Prf}(i-1)+1} \neq d_i$ та $i > 1$:

$$\mathrm{Prf}(i) \leq \mathrm{Prf}(j) + 1, \text{ де } j = \mathrm{Prf}(i-1).$$

Доведемо це висловлювання. Спочатку розглянемо випадок $j > 0$.

З нерівності $d_{\mathrm{Prf}(i-1)+1} \neq d_i$ випливає, що $\mathrm{Prf}(i) \neq \mathrm{Prf}(i-1) + 1$, тобто $\mathrm{Prf}(i) \leq \mathrm{Prf}(i-1) = j$. Отже, послідовність $d_{i-\mathrm{Prf}(i)+1}\, d_{i-\mathrm{Prf}(i)+2} \ldots d_{i-1}$ коротша за $d_{i-j}\, d_{i-j+1} \ldots d_{i-1}$ і є закінченням останньої. З рівності $j = \mathrm{Prf}(i-1)$ та означення префікс-функції маємо:

$$d_1\, d_2 \ldots d_j = d_{i-j}\, d_{i-j+1} \ldots d_{i-1}.$$

Тому збігаються й останні $\mathrm{Prf}(i) - 1$ членів цих послідовностей:

$$d_{j-\mathrm{Prf}(i)+1}\, d_{j-\mathrm{Prf}(i)+2} \ldots d_j = d_{i-\mathrm{Prf}(i)+1}\, d_{i-\mathrm{Prf}(i)+2} \ldots d_{i-1}. \quad (1)$$

З іншого боку, згідно з означенням префікс-функції, маємо:

$$d_1\, d_2 \ldots d_{\mathrm{Prf}(i)} = d_{i-\mathrm{Prf}(i)+1}\, d_{i-\mathrm{Prf}(i)+2} \ldots d_i.$$

Відкинувши останній символ, отримаємо:

$$d_1\, d_2 \ldots d_{\mathrm{Prf}(i)-1} = d_{i-\mathrm{Prf}(i)+1}\, d_{i-\mathrm{Prf}(i)+2} \ldots d_{i-1}. \quad (2)$$

Поєднання (1) та (2) дає:

$$d_1\, d_2 \ldots d_{\mathrm{Prf}(i)-1} = d_{i-\mathrm{Prf}(i)+1}\, d_{i-\mathrm{Prf}(i)+2} \ldots d_{i-1} = d_{j-\mathrm{Prf}(i)+1}\, d_{j-\mathrm{Prf}(i)+2} \ldots d_j.$$

Остання рівність означає: $\mathrm{Prf}(i) - 1$ «претендує» на роль префікс-функції від j. Тобто $\mathrm{Prf}(j)$ не менше за $\mathrm{Prf}(i) - 1$, або ж $\mathrm{Prf}(i) \le \mathrm{Prf}(j) + 1$, що й треба було довести.

Тепер доведемо це ж саме висловлювання для $j = 0$. При $\mathrm{Prf}(i - 1) = j = 0$ нерівність $d_{\mathrm{Prf}(i-1)+1} \ne d_i$ набирає вигляду $d_1 \ne d_i$. Звідси $\mathrm{Prf}(i) \ne 1$. З раніше доведеної нерівності $\mathrm{Prf}(i) \le \mathrm{Prf}(i - 1) + 1$ маємо $\mathrm{Prf}(i) \le j + 1 = 1$. Залишається лише варіант $\mathrm{Prf}(i) = 0$, звідки $\mathrm{Prf}(i) \le \mathrm{Prf}(j) + 1$.

При $\mathrm{Prf}(i - 1) = j > 0$:

- у випадку $d_{\mathrm{Prf}(j)+1} = d_i$, згідно з доведеним висловлюванням, справджується рівність $\mathrm{Prf}(i) = \mathrm{Prf}(j) + 1$;
- у випадку $d_{\mathrm{Prf}(j)+1} \ne d_i$ ми аналогічно доводимо нерівність $\mathrm{Prf}(i) \le \mathrm{Prf}(j') + 1$, де $j' = \mathrm{Prf}(j)$. І якщо $j' > 0$, то переходимо до порівняння символів $d_{\mathrm{Prf}(k)+1}$ та d_i. Так ми будемо замінювати j на $\mathrm{Prf}(j)$, поки не отримаємо $d_{\mathrm{Prf}(j)+1} = d_i$ або не зупинимося на $j = 0$, звідки, як вже доведено, випливатиме $\mathrm{Prf}(i) = 0$.

В обох випадках $\mathrm{Prf}(i) = \mathrm{Prf}(j) + 1$ – саме для цього ми поклали $\mathrm{Prf}(0) = -1$.

Для кожного i у випадку $d_{\mathrm{Prf}(i-1)+1} = d_i$ значення $\mathrm{Prf}(i)$ досягає теоретичного максимуму, який дорівнює $\mathrm{Prf}(i - 1) + 1$. Кожна додаткова перевірка рівності $d_{\mathrm{Prf}(j)+1}$ і d_i приводить до зменшення максимально можливого значення $\mathrm{Prf}(i)$ хоча б на 1, бо $\mathrm{Prf}(j) \le j - 1$. Маємо:

$\mathrm{Prf}(i) \le \mathrm{Prf}(i - 1) + 1 - (p_i - 1)$,

де p_i — загальна кількість перевірок рівності елементів при даному i. Звідси:

$p_i \le \mathrm{Prf}(i - 1) - \mathrm{Prf}(i) + 2$.

Додавши праві й ліві частини нерівностей, отримаємо:

$p_1 + \ldots + p_{n-1} \le \mathrm{Prf}(0) - \mathrm{Prf}(n - 1) + 2(n - 1) < 2n$.

Таким чином, кількість операцій залежить від n лінійно.

Тепер для довільної послідовності $s_1 s_2 \ldots s_m$ при всіх $i = 1, 2, \ldots, m$ можна визначити найбільше $k \le i$, при якому $s_{i-k+1} s_{i-k+2} \ldots s_i = d_1 d_2 \ldots d_k$. Для цього достатньо у попередній процедурі замінити перевірки $d_{\mathrm{Prf}(j)+1} = d_i$ на $d_{\mathrm{Prf}(j)+1} = s_i$. Тоді при $i = m$ відповідне k дорівнює максимальній довжині послідовності вигляду $d_1 d_2 \ldots d_i$, що є закінченням $s_1 s_2 \ldots s_m$. Наступна за довжиною послідовність $d_1 d_2 \ldots d_i$, що є закінченням $s_1 s_2 \ldots s_m$, матиме довжину $\mathrm{Prf}(k)$, потім — $\mathrm{Prf}(\mathrm{Prf}(k))$ і так далі.

Якщо за $s_1 s_2 \ldots s_m$ взяти $d_{n-1} d_{n-2} \ldots d_1$, то таким чином ми знайдемо всі симетричні рядки виду $d_1 d_2 \ldots d_i$.

Є простіший підхід до цієї задачі, який дуже схожий на алгоритм Рабіна —Карпа. Але цей алгоритм спирається на випадковість.

Потрібно обрати достатньо велике натуральне m — наприклад, порядку 40000, щоб m^2 не перевищувало максимального значення типу longint. Потім потрібно обрати випадкове натуральне $a < m$ і для кожного $i = 1, 2, \ldots, n - 1$ порахувати

$$h(i) = (d_1 + d_2 a + \ldots + d_i a^{i-1}) \pmod m.$$

Це легко зробити за допомогою динамічного програмування, бо

$$h(i) = (h(i - 1) + d_i \cdot (a^{i-1} \bmod m)) \pmod m,$$

а величину $(a^{i-1} \bmod m)$ можна вираховувати послідовно для $i = 1, 2, \ldots, n - 1$. Потім для кожного i знайдемо

$$g(i) = (d_i + d_{i-1} a + \ldots + d_1 a^{i-1}) \pmod m,$$

скориставшись співвідношенням:

$$g(i) = (d_i + g(i - 1) \cdot a) \pmod m.$$

Для симетричних послідовностей $d_1 d_2 \ldots d_i$ завжди справджується рівність: $h(i) = g(i)$. Для несиметричних послідовностей ця рівність справджується з імовірністю (приблизно) $1/m$. Тому безпосередньо перевіряючи на симетричність ті послідовності

$d_1\, d_2 \ldots d_i$, для яких $h(i) = g(i)$ і справджується аналогічний критерій симетричності для $d_{i+2}\, d_{i+3} \ldots d_{n-1}$, ми швидко знайдемо розбиття на справді симетричні послідовності.