Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №7
По дисциплине « ***Алгоритмы и структуры данных*** »
Тема: ***«Префиксные деревья»***

Выполнил:
Студент 2 курса
Группы ПО-11(2)
Сымоник И.А
Проверила:
Глущенко Т.А

Брест 2023

Цель работы: изучить префиксные деревья их применение и реализацию.

Ход работы

Задание 1. Реализовать *префиксное* дерево, задача *208* ресурса *https://leetcode.com /*.

Исходный код:

```
class Trie
{
 public:
   Trie()
   {
      root=new Node();
   }

   void insert(string word)
   {
      Node* temp=root;
      for(int i=0;i<word.size();i++)
      {
         if(!temp->Contains(word[i]))
            temp->Put(word[i],new Node());
         temp=temp->Get(word[i]);
      }
      temp->SetFlag();
   }

   bool search(string word)
   {
      Node* temp=root;
      for(int i=0;i<word.size();i++)
      {
         if(!temp->Contains(word[i]))
            return false;
         temp=temp->Get(word[i]);
      }
      return temp->IsFlagSet();
   }

   bool startsWith(string prefix)
   {
      Node* temp=root;
      for(int i=0;i<prefix.size();i++)
      {
         if(!temp->Contains(prefix[i]))
            return false;
         temp=temp->Get(prefix[i]);
      }
      return true;
```

```
    }

    private:


    struct Node{
       Node* links[26];
       bool flag = false;

       bool Contains(char ch)
       {
          return (links[ch-'a']!=NULL);
       }
       void Put(char ch,Node *temp)
       {
          links[ch-'a']=temp;
       }
       Node* Get(char ch)
       {
          return links[ch-'a'];
       }
       void SetFlag()
       {
          flag=true;
       }
       bool IsFlagSet()
       {
          return flag;
       }
    };
    Node* root;

};
```

Результат:

Runtime
41ms
Beats 92.17%of users with C++
Memory
43.82MB
Beats 59.54%of users with C++

Задание 2. Решить задачу **142. Linked List Cycle II** с ресурса leetcode.

Исходный код:

```
class Solution {
```

```
public:
   ListNode *detectCycle(ListNode *head) {

      ListNode *fast = head;
      ListNode *slow = head;

      while(fast != nullptr && fast->next != nullptr)
      {
         slow = slow->next;
         fast = fast->next->next;
         if(slow == fast)
         {
            slow = head;
            while(slow != fast)
            {
               fast = fast->next;
               slow = slow->next;
            }
            return slow;
         }
      }
      return nullptr;
   }
};
```

Результат:

Runtime

3ms

Beats 97.58%of users with C++

Memory
7.92MB
Beats 71.72%of users with C++

Задание 3. Решить задачу **287. Find the Duplicate Number** с ресурса leetcode.

Исходный код:

```
class Solution {
public:
   int findDuplicate(vector<int>& nums)
   {
      int slow = nums[0];
      int fast = nums[nums[0]];
      while(slow != fast)
      {
```

```
            slow = nums[slow];
            fast = nums[nums[fast]];
        }

        slow = 0;
        while(slow != fast)
        {
            slow = nums[slow];
            fast = nums[fast];
        }

        return slow;
    }
};
```

Результат:

Runtime
78ms
Beats 79.74%of users with C++
Memory
61.62MB
Beats 38.17%of users with C++

Контрольные вопросы

1. Корректен ли алгоритм, если имеются *два* и более дубликата?

   Алгоритм не корректен для 2 и более дубликатов, так как из количество заранее не известно, следовательно неизвестно сколько раз отнимать максимальное число.

2. Какие варианты решения задачи еще существуют, если есть ограничения только по временной сложности?

   Полный перебор (Brute force) – $O(n^2)$
   При помощи сортировки – O(n*logn)
   При помощи быстрого и медленного указателя – O(n)

3. Какова временная сложность решения данной задачи методом *brute force*

   Сложность решения полным перебором составляет $O(n^2)$

4. Что хранится в узле префиксного дерева?

Массив указателей на следующие узлы и флаг, который указывает был ли символ конечным.

5. Чем определяется ключ, соответствующий узлу дерева?
   Ключ определяется конкатенацией предыдущих и текущего узлов.

6. От чего зависит время поиска в префиксном дереве?

   Время поиска в префиксном дереве зависит от длины ключа и размера алфавита

7. Перечислите основные операции для префиксных деревьев и их временную сложность.

   Операция поиска – O(n)
   Операция вставки – O(n)
   Операция удаления - O(n), где n – длина ключа.

8. Где применяются префиксные деревья?

1. Хранение словарей и автозаполнение: префиксные деревья часто используется для эффективного хранения больших словарей, таких как английский словарь, и реализации автозаполнения в поисковых системах и редакторах текста.

2. Компиляторы и анализаторы: префиксные деревья могут использоваться в компиляторах и анализаторах для построения таблицы ключевых слов, идентификаторов и других символьных последовательностей.

3. Компьютерные сети: префиксные деревья используются в радиотехнике и сетевых протоколах для поиска наилучшего соответствия адресов, также известного как longest prefix match. Это важно, например, в IP-маршрутизации.

4. Поиск по префиксу (Prefix Matching): префиксные деревья эффективно поддерживает операции поиска по префиксу, что делает его полезным для поиска слов в словарях, автозаполнения при вводе текста и

других приложений, где необходимо быстро находить слова по начальной части.

9. Что такое метод *декомпозиции* и в каких алгоритмах он применяется?

Метод декомпозиции — это подход к решению сложных задач, который заключается в разбиении задач на более простые подзадачи. Это позволяет более эффективно решать сложную задачу, так как каждая подзадача может быть решена независимо, а затем объединена для получения окончательного результата. Метод применяется в :

- Бинарном поиске.
- Сортировке слиянием.
- Быстрой сортировке.
- Умножение матриц методом Штрассена.
- Алгоритме Карацубы.

Вывод: Изучили префиксные деревья и способы работы с ними.