**Урок по теме: «Двоичный поиск» в 10 классе**

**учителя информатики и ИКТ МБОУ ФМЛ Барулиной Н.Н.**

**Тема урока:** «**Двоичный поиск**»

**Тип урока:** изучение нового материала.

**Вид урока:** комбинированный.

**Оборудование:**

* программное обеспечение: АВС Паскаль;
* мультимедийный проектор.

**Цели урока:**

*Образовательная:*

* Приобретение знаний и навыков двоичного поиска элемента в массиве.
* Закрепление умения сортировки массива.
* Развитие навыков работы в коллективе, умений излагать изученный материал.

*Развивающая:*

* Развитие алгоритмического мышления. Научить выделять главное при решении задач.
* Формирование и развитие функционального мышления учащихся, развитие познавательных потребностей учащихся, создание условий для приобретения опыта работы учащихся в среде ИКТ.

*Воспитательная:*

* Воспитание познавательного интереса к предмету информатика с помощью практических заданий.
* Развитие самостоятельности, внимательности, аккуратности, умения работать индивидуально по заданной теме.

**Обучающие технологии:**

* ИКТ;
* Проблемное обучение;
* Технологии уровневой дифференциации ;
* Здоровьесберегающие.

План урока:

1. Орг. Момент (1 мин)
2. Актуализация опорных знаний. Повторение способов сортировки массивов (7 мин)
3. Объяснение нового материала (15 мин)
4. Выполнение задания (20)
5. Итог урока (2 мин).

Слайд 1 Тема урока «Двоичный поиск».

Сегодня на уроке мы узнаем, что это за способ, для чего он применяется

На уроках мы с вами познакомились с некоторыми способами сортировки массивов. Какие сортировки вы знаете ? (метод пузырька (сортировка обменами), простой выбор, быстрая сортировка)

Фрагмент какой сортировки представлен на слайде?

Найдите ошибки

|  |  |
| --- | --- |
| Текст слайда (слайд 2) | Правильный вариант (слайд 3) |
| for i:= 1 to N-1 do  for j:= N-1 to i do  if A[j+1]< A[j] then begin  L:=A[j+1];  A[j+1]:= A[j];  L:= A[j]  end;  («Пузырек») | for i:= 1 to N-1 do  for j:= N-1 downto i do  if A[j+1]< A[j] then begin  L:=A[j+1];  A[j+1]:= A[j];  A[j]:=L  end; |
| Слайд 4  for i:= 1 to N-1 do begin  nMin:= 1;  for j:=i+1 to N do  if A[j] < A[nMin] then  nMin:= j;  if i <> nMin then  L:=A[j+1];  A[j+1]:= A[j];  A[j]:=L  end;  (метод выбора) | Слайд 5  for i:= 1 to N-1 do begin  nMin:= i;  for j:=i+1 to N do  if A[j] < A[nMin] then  nMin:= j;  if i <> nMin then begin  L:=A[j+1];  A[j+1]:= A[j];  A[j]:=L  end;  end; |

- Какой из данных методов требует меньше перестановок? (Метод выбора).

- Для чего нужна сортировка? (Чтобы облегчить дальнейший поиск элементов)

Ранее мы рассматривали задачу поиска элемента массива, равного данному.

- Как можно организовать данный поиск? (Сравнить все элементы массива с заданным числом).

Такой поиск называется линейным.

- Сколько сравнений мы делаем для массива из 1000 элементов? (1000 сравнений, чтобы убедиться что такого элемента нет). Если же количество элементов велико, то время поиска может оказаться не допустимым.

- А можно ли уменьшить количество операций, если мы имеем уже отсортированный массив?

Если у нас есть массив, содержащий упорядоченную последовательность данных, то очень эффективен двоичный (бинарный) поиск. Слайд 6

Метод использует стратегию «разделяй и властвуй», а именно: заданная последовательность делится на две равные части и поиск осуществляется в одной из этих частей, которая потом также делится надвое, и так до тех пор, пока обнаружится наличие искомого элемента или его отсутствие.

- Сколько операций сравнения потребуется для поиска числа в массиве из 1023 элементов? (10)

- Как организовать данный поиск?

Слайд 7

1. Выбрать средний элемент A[c] и сравнить с X.
2. Если X = A[c], то нашли (стоп).
3. Если X < A[c], искать дальше в первой половине.
4. Если X > A[c], искать дальше во второй половине.

Напишем фрагмент кода программы для организации бинарного поиска

Код программы пишет на доске ученик.

Вводим необходимые обозначения: L левая граница рассматриваемого интервала

R – правая . С-номер среднего элемента.

Как найти номер среднего элемента? ((L+R) div 2)

Какой цикл будем использовать? (цикл пока, неизвестно количество выполнений цикла)

Как правильно поставить условие для входа в цикл?

После написания кода программы осуществляется трассировка для числа, находящегося в массиве, для числа, отсутствующего в массиве, для чисел, стоящих на крайних позициях

L:=0; R:=N+1;

WHILE L<R-1 do begin

C:=(R+L) div 2;

if a[c]=x then break;

if a[c]<x then L:=C else R:=C;

end;

if a[c]<> x then writeln ('элемент не найден') else writeln (**('A[',c,']=',X)**;

Задачи . Выполняются учащимися за компьютером.

1. Заполнить массив из 20 элементов по формуле a[i]:=3\*i-1. Ввести число X. Используя двоичный поиск, определить, есть ли в массиве число, равное X. Подсчитать количество сравнений. На экран вывести элементы массива, сообщение о номере найденного элемента или об отсутствии числа и количество сравнений.
2. Заполнить массив случайными числами и отсортировать его. Ввести число X. Используя двоичный поиск, определить, сколько чисел, равных X, находится в массиве.

|  |  |
| --- | --- |
| **Пример:**  **Массив:**  **1 4 7 3 9 2 4 5 2**  **После сортировки:**  **1 2 2 3 4 4 5 7 9**  **Введите число X:**  **4**  **Число 4 встречается 2 раз(а).** | **Пример:**  **Массив:**  **1 4 7 3 9 2 4 5 2**  **После сортировки:**  **1 2 2 3 4 4 5 7 9**  **Введите число X:**  **14**  **Число 14 не встречается.** |

1. Заполнить массив случайными числами и ввести число и отсортировать его. Ввести число X. Используя двоичный поиск, определить, есть ли в массиве число, равное X. Если такого числа нет, вывести число, ближайшее к X.

|  |  |
| --- | --- |
| **Пример:**  **Массив:**  **1 4 12 3 19 2 4 5 2**  **После сортировки:**  **1 2 2 3 4 4 5 12 19**  **Введите число X:**  **12**  **Число 12 найдено.** | **Пример:**  **Массив:**  **1 4 12 3 19 2 4 5 2**  **После сортировки:**  **1 2 2 3 4 4 5 12 19**  **Введите число X:**  **11**  **Число 11 не найдено.**  **Ближайшее число 12.** |

Подведение итогов.

Сегодня на уроке мы познакомились с Двоичным поиском в массиве. Когда этот способ удобно использовать? (Так как данные для этого способа сначала надо отсортировать, а это может занять больше времени, чем сам поиск. Поэтому такой подход эффективен, если данные меняются и сортируются редко, а поиск выполняется часто)

**Домашнее задание**

§67 Написать программы

Используемые материалы:

http://kpolyakov.spb.ru/school/probook/prakt.htm