СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ ПОИСКА

Ж. Г. Лазовская

Барановичский государственный университет
Барановичи, Беларусь
E-mail: zhann_ka@mail.ru

Подготовка учителей информатики требует от студентов глубоких теоретических сведений из теории алгоритмов. Цель решения той или иной задачи студентом состоит не только в реализации составленного алгоритма на компьютере, но и в его анализе в целях нахождения эффективного алгоритма. Одним из важных типов задач, встречающихся в вычислительной технике, является поиск. Программная реализации трех методов поиска в упорядоченном массиве – бинарный, фибоначчиев и интерполяционный, позволит глубже уяснить понятие эффективного алгоритма.

Ключевые слова: алгоритм, эффективный алгоритм, бинарный поиск, поиск Фибоначчи, интерполяционный поиск.

Анализ публикаций о преподавании информатики в школе и в высших учебных заведениях позволяет говорить о фундаментализации обучения этой дисциплине. Так, в своей работе «О фундаментализации подготовки учителя информатики» А. И. Павловский, В. К. Пономаренко пишут: «Значит, фундаментализация обучения информатике вместе с главной ролью ее научной базы предусматривает основательность, глубину и прочность знаний по предмету. Но чтобы добиться этого, необходимо пробудить у учащихся интерес к теоретическим аспектам дисциплины. Мы считаем, что достижению этой цели наилучшим образом будет способствовать совокупность удачно подобранных задач. Именно решение задач с получением разных алгоритмов решения одной и той же задачи и выбором среди них наилучшего позволит подвести студентов к идее о необходимости оценивания сложности алгоритма и использовании теории вычислительной сложности алгоритмов при решении реальных научных и производственных задач» [6].

Таким образом, студент педагогического вуза, будущий учитель информатики, должен иметь представление о стандартном наборе основных алгоритмов, относящихся к разным областям вычислительной техники, уметь разрабатывать алгоритмы и анализировать их эффективность.

На практике существуют два вида оценки эффективности алгоритма: временная и пространственная. Временная эффективность является индикатором скорости работы алгоритма, а пространственная эффективность показывает количество дополнительной оперативной памяти, необходимой для работы алгоритма. Алгоритмы, разработанные для решения одной и той же задачи, часто очень сильно различаются по эффективности. Эти различия могут быть намного значительнее, чем те, что вызваны применением неодинакового аппаратного и программного обеспечения.

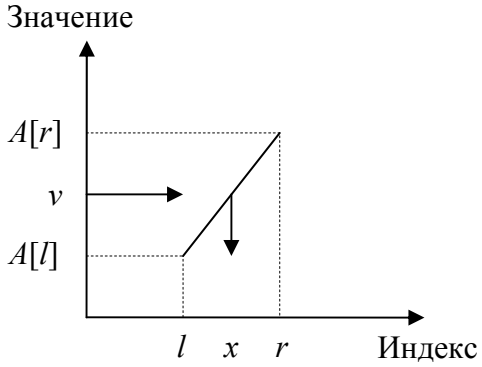
Среди огромного количества задач, встречающихся в вычислительной технике, можно выделить несколько типов, которым ученые всегда уделяли особое внимание. Подобный интерес вызван либо практическим значением задачи, либо какими-то ценными ее свойствами, представляющими особый интерес для исследования.

Одним из важных типов задач является поиск. Задача поиска связана с нахождением заданного значения, называемого ключом поиска, среди заданного множества. Существует огромное количество алгоритмов поиска. Их сложность варьируется от самых простых алгоритмов поиска методом последовательного сравнения, до чрезвычайно эффективных, но ограниченных алгоритмов бинарного поиска, а также алгоритмов, основанных на представлении базового множества в иной, более подходящей для выполнения поиска форме. Для решения задачи поиска также не существует единого алгоритма, который бы наилучшим образом подходил для всех случаев. Некоторые из алгоритмов выполняются быстрее остальных, но для их работы требуется дополнительная оперативная память. Другие выполняются очень быстро, но их можно применять только для предварительно отсортированных массивов и так далее.

Рассмотрим три алгоритма поиска в отсортированном массиве: двоичный (бинарный поиск), фибоначчиев и интерполяционный поиск. Необходимо отметить, что все три метода подробно описаны и проанализированы в [2].

Бинарный поиск представляет собой эффективный алгоритм для поиска в отсортированном массиве. Он работает путем сравнения искомого ключа K со средним элементом массива $A[m]$. Если они равны, алгоритм прекращает работу. В противном случае та же операция рекурсивно повторяется для первой половины массива, если $K < A[m]$, и для второй, если $K > A[m]$.

Еще одним примером алгоритма поиска в отсортированном массиве является интерполяционный поиск. Интерполяционный поиск учитывает значение ключа поиска при определении элемента массива, который будет сравниваться с ключом. При выполнении итерации поиска между элементами A[l] (крайним слева) A[r] (крайним справа), алгоритм предполагает, что значения в массиве растут линейно. В соответствии с этим предположением, значение v ключа поиска сравнивается с элементом, индекс которого вычисляется (с округлением) как координата x точки на прямой, проходящей через (l, $A[l]$) (r, $A[r]$), координата y которой равна значению v (рисунок).



Вычисление индекса
при интерполяционном поиске

Записав стандартное уравнение для прямой, проходящей через две точки (l, $A[l]$) (r, $A[r]$), заменив в нем y на v и решая его относительно x, получим формулу:

$$x = l + \left\lfloor \frac{(v - A[l])(r - l)}{A[r] - A[l]} \right\rfloor. \qquad (1)$$

Значения массива возрастают (точнее, не убывают) от A[l] до A[r]. Пусть это возрастание – линейное, вычисленное по формуле (1) значение индекса – ожидаемая позиция элемента со значением, равным v. После сравнения v с $A[x]$ алгоритм либо прекращает работу

(если они равны), либо продолжает поиск тем же способом среди элементов с индексами, либо от l до x–1, либо от x+1 до r, в зависимости от того, меньше ли v значения $A[x]$ или больше.

Алгоритм поиска ключевого значения K в отсортированном массиве A [1…N] методом Фибоначчи можно описать следующим псевдокодом (для удобства описания предполагается, что N+1 представляет собой число Фибоначчи F_{k+1}):

1. Установить $i \leftarrow F_k$, $p \leftarrow F_{k-1}$, $q \leftarrow F_{k-2}$ // p и q означают последовательные числа Фибоначчи.

2. Если $K < A[i]$, перейти к шагу 3; если $K > A[i]$, перейти к шагу 4; если $K = A[i]$, то алгоритм успешно завершается.

3. Если $q = 0$, алгоритм завершается неудачно. В противном случае следует установить $I \leftarrow i–q$ и $(p, q) \leftarrow (q, p–q)$; затем перейти к шагу 2.

4. Если $p = 1$, алгоритм завершается неудачно. В противном случае следует установить сначала $i \leftarrow i+q$ и $p \leftarrow p–q$, а затем – $q \leftarrow q–p$ и перейти к шагу 2.

Описанные в данном исследовании три метода поиска были реализованы в системе программирования Pascal ABC.

Для проведения экспериментальных исследований была поставлена задача: пусть имеется некоторый целочисленный массив A, элементами которого являются случайные числа. Данный массив сортируем по возрастанию, задаем ключ для поиска. Для каждого из трех описанных методов поиска ключевого значения предусматривается вывод искомого элемента, его индекс, количество итераций. В ходе решения задачи были составлены два алгоритма Poisk и Poisk1, данные алгоритмы были протестированы для массивов размерностью 100, 500 и 1000.

Отличием алгоритма Poisk от алгоритма Poisk1 является равномерность возрастания элементов массива. Ниже приведены фрагменты программ Poisk и Poisk1 по заполнению целочисленного массива A размерностью 100.

```
begin
n:=100;
// заполняем массив
// будем искать значение элемента
массива, расположенного в k-й ячейке
writeln('Исходный массив');
for i:=1 to round(n/2) do begin
A[i]:=random(100);
write(A[i],';');
end;
for i:=round(n/2)+1 to n do begin
A[i]:=random(900000)+100;
write(A[i],';');
end;
```

```
begin
n:=100;
// заполняем массив
// будем искать значение элемен-
та массива, расположенного в k-й
ячейке
writeln('Исходный массив');
for i:=1 to n do begin
A[i]:=random(1000000);
write(A[i],';');
end;
```

Было замечено, что в случае равномерного возрастания массива интерполяционный поиск оказывается наиболее удачным – количество итераций в большинстве случаев оказывается наименьшим (в случаях размерности массива 500 и 1000 данное замечание также является верным).

Отобразим результаты выполнения программ в таблице. Анализ приведенных данных позволяет утверждать, что в конкретном экземпляре задачи наиболее выгодным является интерполяционный поиск (при условии равномерного возрастания элементов массива, и, чем больше n, тем это преимущество очевидней – алгоритм Poisk1). В алгоритме Poisk наиболее выгодным является бинарный метод. Количество итераций поиска Фибоначчи оказывается наибольшими во всех испытаниях алгоритма Poisk1, и сопоставим по эффективности с бинарным поиском в алгоритме Poisk.

№ п/п	Poisk									Poisk1								
	Размерность массива $n = 100$			Размерность массива $n = 500$			Размерность массива $n = 1000$			Размерность массива $n = 100$			Размерность массива $n = 500$			Размерность массива $n = 1000$		
	Количество итераций																	
	Бинарный	Фибонач-чиев	Интерпо-ляционный	Бинарный	Фибонач-чиев	Интерпо-ляционный	Бинарный	Фибонач-чиев	Интерпо-ляционный	Бинарный	Фибонач-чиев	Интерпо-ляционный	Бинарный	Фибонач-чиев	Интерпо-ляционный	Бинарный	Фибонач-чиев	Интерпо-ляционный
1	8	10	5	4	9	64	9	11	98	4	5	1	9	10	3	11	15	4
2	7	9	3	9	13	78	11	10	69	7	9	4	10	13	2	10	14	2
3	8	4	4	8	13	5	9	14	84	8	7	2	10	12	2	8	13	3
4	6	7	4	7	11	72	6	12	95	8	8	3	9	13	3	10	14	3
5	8	10	6	8	13	68	10	14	3	6	10	3	9	13	1	10	11	4
6	8	10	14	10	13	65	11	11	98	7	8	2	10	10	2	9	15	2
7	7	8	6	10	11	12	11	13	46	4	8	2	9	9	4	11	15	4
8	7	10	13	10	8	14	11	12	6	5	10	2	9	10	5	11	15	3
9	8	8	6	8	9	29	8	15	41	7	9	2	9	12	2	10	15	4
10	7	9	16	8	13	5	10	10	69	3	9	3	5	11	2	11	12	1

Данное исследование не содержит аналитическую основу анализа временной эффективности алгоритмов. Однако полученные практические результаты позволят уделить внимание методикам проектирования алгоритмов, выбору альтернативных методов решения типовых задач, а значит, усилить теоретическую подготовку учителей информатики.

ЛИТЕРАТУРА

1. *Бейда, А. А.* Задачи по информатике с элементами доказательности / А. А. Бейда, А. И. Павловский // Информатизация образования. 2002. № 2. С. 28–45.
2. *Кнут, Д. Э.* Искусство программирования / Д. Э. Кнут. 2-е изд. ; пер. с англ. М. : Вильямс, 2007. Т. 3: Сортировка и поиск. 832 с.
3. Алгоритмы: построение и анализ / Т. Х. Кормен [и др.]. 2-е изд. ; пер. с англ. М. : Вильямс, 2007. 1296 с.
4. *Левитин, А. В.* Алгоритмы: введение в разработку и анализ / А. В. Левитин ; пер. с англ. М. : Вильямс, 2006. 576 с.
5. *Павловский, А. И.* Теоретические основы алгоритмизации: пособие / А. И. Павловский, В. В. Пенкрат. Минск : БГПУ, 2007. 60 с.
6. *Паўлоўскі, А. І.* Аб фундаментальнай падрыхтоўкі настаўніка інфарматыкі / А. І. Паўлоўскі, В. К. Панамарэнка // Вес. БДПУ. Сер. 3. 2004. № 1. С. 22–24.