Изучаване на алгоритми за търсене с помощта на анимации

Цветозар Георгиев

Learning search algorithms through animations: *The paper looks at the implementation of a multimedia application that allows students to study search algorithms using animations. The algorithms which are included in the application are: sequential search, sequential search with reordering, binary search, quadratic search and interpolation search. The application enables students to understand not only how search algorithms work, but also to compare their complexity and performance.*

Key words: *Programming, Search Algorithms.*

ВЪВЕДЕНИЕ

При изучаването на алгоритми използването на анимации се налага все повече като предпочитан метод. Предимствата на това нагледно представяне са няколко [1, 3]: по-лесно усвояване на алгоритмите, благодарение на визуализацията на скритите стъпки; възможност на потребителя да експериментира с различни данни както и възможност за многократно повторение на анимациите.

От извършения преглед на съществуващите реализации за нагледно представяне на алгоритми за търсене могат да се направят следните изводи:

- съществуват реализации основно на три алгоритъма – последователно търсене [1, 4, 5], двоично търсене [1, 6, 7] и интерполационно търсене [1];
- те са реализирани във вид на Java аплети;
- потребителският интерфейс на всички е на английски език;
- реализациите не са съпроводени с теоретичен материал;
- потребителят не може да въвежда последователности от стойности [4, 6, 7] или има изисквания въведените стойности да са малко на брой и да са в нарастващ ред [5];
- потребителят няма възможност да сравнява сложността и да оценява бързодействието на различните алгоритми.

ИЗЛОЖЕНИЕ

За решаване на тези задачи, посредством средата Adobe Flash и езика за програмиране ActionScript беше разработено мултимедийно приложение, което може да се използва за обучение и самообучение на студенти за най-популярните алгоритми за търсене.

Алгоритмите за търсене, които са реализирани в приложението са:

- Последователно търсене [2]. При този алгоритъм целият масив се обхожда последователно като се търси номера на елемента, който съвпада с търсения.
- Последователно търсене с преподреждане. При този алгоритъм елементите на масива са подредени според своите броячи. Всеки брояч пази информация за това, колко пъти дадена стойност е била търсена. По този начин елементът, чиято стойност на брояча е най-голяма е първи, следва този, за когото броячът е най-голям спрямо останалите броячи и т.н. Обхождането също се осъществява последователно.
- Квадратично търсене. При този алгоритъм се избира стъпка k и последователно се извършва проверка дали търсеният елемент x е по-голям от първия елемент в масива, от (k+1)-ия елемент, от (2k+1)-ия елемент, от (3k+1)-ия елемент и т.н. Процесът приключва при достигане до елемент, по-голям или равен на търсения x, или при достигане края на масива. Масивът трябва да бъде подреден и да не се съдържат повтарящи се елементи.
- Двоично търсене [2]. Методът е рекурсивен. Масивът трябва да е подреден. Първоначално търсената стойност се сравнява с ключа на средния елемент в масива. Ако елементът е по-голям, търсенето продължава в дясната част, в

противен случай – в лявата.

- Интерполационно търсене. Този алгоритъм представлява модификация на алгоритъма за двоично търсене. Масивът се разделя на две равни части и индексът m на сравнявания елемент key се изчислява по формулата:

$$m = lf + k * (rt - lf) \quad (1)$$

където:

$$k = \frac{key - a[lf]}{a[rt] - a[lf]} \quad (2)$$

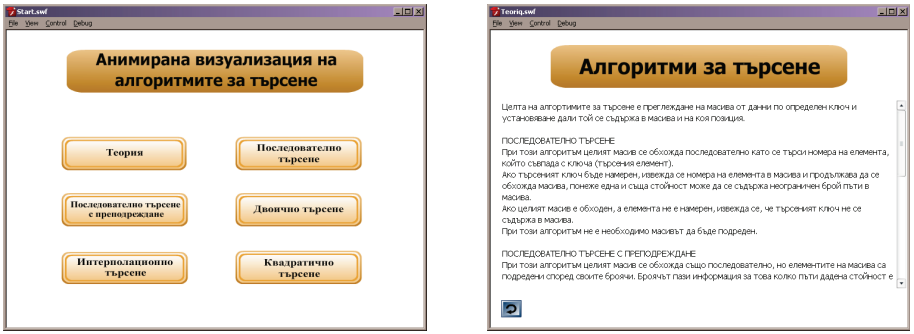
lf е началният индекс в интервала;

rt е крайният индекс в интервала.

Коефициентът k замества константата 1/2 (в двоичното търсене) и дава приблизителната позиция на търсения елемент в разглеждания интервал. Масивът трябва да бъде подреден и да не съдържа повтарящи се елементи. Алгоритъмът е подходящ при големи обеми данни и големи по размер ключове, когато сравненията отнемат много време.

Избраните за реализация в приложението алгоритми за търсене имат еднаква пространствена сложност. Основната разлика между тях е, че при последните три метода, търсенето се прилага върху предварително сортирани данни.

Приложението се състои от начален екран (фиг.1а), посредством който потребителят може да избира дали да премине към екрана с подробна теоретична информация за алгоритмите за търсене (фиг.1б) или към някой от петте модула, където тяхната работа се визуализира с помощта на анимации.



а) б)

Фиг. 1 – Начален екран (а) и екран с теория за алгоритмите за търсене (б)

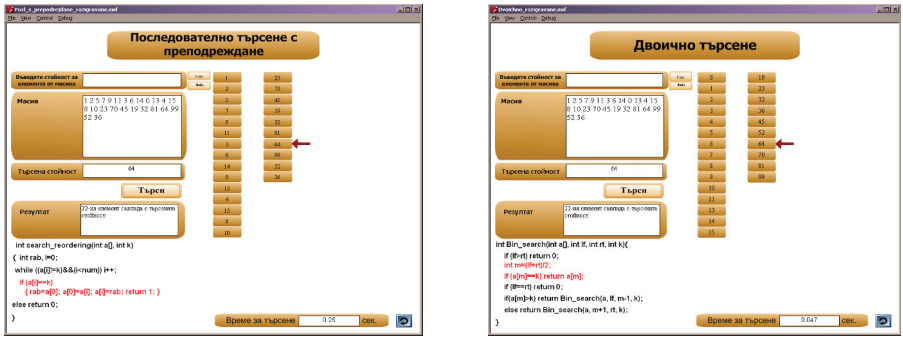
Всеки един от модулите поддържа следните функционални възможности:

- Потребителят може да въвежда, както отделни стойности за елементи на масив, така и да зарежда такива стойности от предварително подготвен текстов файл. При конкретната реализация стойностите в текстовия файл трябва да са разделени със запетая. Предвидена е възможност за смесено въвеждане, при което числата, които се въвеждат ръчно се разполагат в началото на масива, преди тези, които се зареждат от файл. Във всеки от модулите е предвидено поле за въвеждане на стойност на елемент от масива и бутон „Enter”, чрез който се осъществява тази операция. Бутонът “Файл” служи за прочитане на текстов файл с данни. В полето „Масив” се визуализират въведените от потребителя числени стойности. Ако потребителят въведе символи, различни от цифри или въведе повтарящи се числа се извеждат предупредителни съобщения. С цел добра визуализация максималният брой на елементите е ограничен до 60.

- При метода „Квадратично търсене” (фиг.3б) потребителят, чрез полето

„Стъпка", трябва да зададе стъпката (цяло положително число), с която да се обхождат елементите на масива. Предвиден е логически контрол на въвежданата в полето информация.

- Потребителят има възможност да задава стойност за търсене в полето "Търсена стойност", след което посредством бутона „Търси" се реализира процедурата за търсене. Аналогично на въвеждането на елементи в масива, ако в това поле бъде въведен символ, различен от цифра се извежда предупредително съобщение.



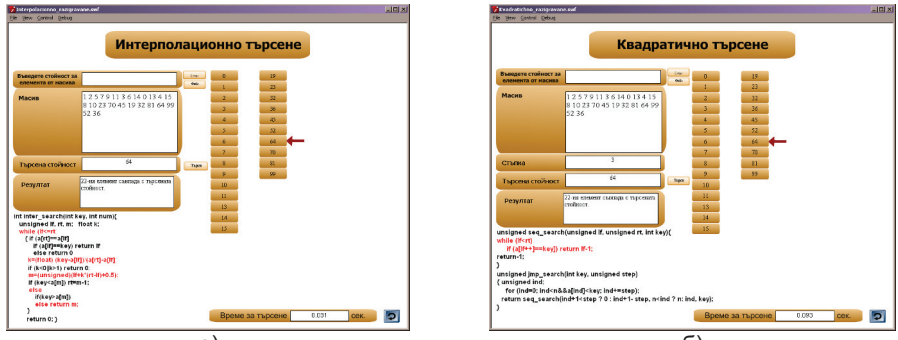
а) б)

Фиг. 2 – Последователно търсене с преподреждане (а) и двоично търсене (б)

- В полето „Резултат" се извежда информация дали е открит елемент със зададената стойност и неговия номер. Ако не бъде открит елемент се извежда съобщение.

- Под полето „Резултат" се извежда кода на съответния алгоритъм за търсене, написан на C++. На всяка стъпка от изпълнението на алгоритъма се оцветява съответната част от кода. Това подпомага потребителя да получи визуална представа какво участие има всяка част от кода в алгоритъма за търсене.

- В дясната част на екраните се визуализират всички елементи на масива, а с помощта на стрелка се анимира обхождането на масива в съответствие с избрания метод за търсене.



а) б)

Фиг. 3 – Интерполационно търсене (а) и квадратично търсене (б)

- В полето „Време за търсене" се извежда относителното време (време за търсене + време за визуализация) в секунди, което е използвал избраният

алгоритъм за намиране на зададена от потребителя стойност. По този начин потребителите могат да извършат сравнение за бързодействието на различните алгоритми при търсене на една и съща стойност в един и същи масив. Например при последователно търсене с преподреждане (фиг.2а) времето за търсене на числото 64 сред 25 елемента е 0.25 секунди. При двоично търсене (фиг.2б) на същото число сред същите елементи времето е 0.047 секунди. При интерполационно търсене (фиг.3а) се получава време 0.031 секунди, а при квадратично търсене (фиг.3б) – 0.093 секунди. От така получените резултати за конкретния случай се получава, че най-бързо е интерполационното търсене, следвано от двоичното и квадратичното.

При три от алгоритмите за търсене (двоично, интерполационно и квадратично) е необходимо масивът да бъде подреден. Съществуват много методи за сортиране, а този, който е използван в проекта е сортиране във възходящ ред чрез пряка селекция. При този метод, в един n-мерен масив, първо се намира най-малкият елемент; след това този елемент си разменя мястото с първия елемент. Тези операции се повтарят с останалите n-1 елемента, след това с останалите n-2 елемента и т.н., докато остане само един елемент – най-големия.

При всеки от петте модула потребителят има възможност да се върне към началния екран, а от там да премине към екрана с теория или към някой от другите модули.

ЗАКЛЮЧЕНИЕ

В резултат на работата е реализирано приложение, което дава възможност за нагледно представяне посредством анимации на някои от най-популярните алгоритми за търсене.

Приложението ще се използва в процеса на обучение на студенти по дисциплината „Синтез и анализ на алгоритми”. То ще даде възможност на студентите да добият представа не само как работят алгоритмите за търсене, но и да сравняват какво е тяхното бързодействие. С негова помощ те ще могат да оценяват и асимптотичната сложност на различните алгоритми за търсене.

Тъй като алгоритмите са реализирани посредством езика за програмиране ActionScript на Flash, то техният код може да се използва при обучението на студенти по дисциплината „Мултимедийни системи и технологии”.

ЛИТЕРАТУРА

[1] Bremananth, R. et al. Visualization of Searching and Sorting Algorithms, International Journal of Computer and Information Engineering, vol. 3, 2009, pp.194-202.

[2] Niemann, T., Sorting and Searching Algorithms, http://epaperpress.com/sortsearch/download/sortsearch.pdf

[3] Wilson, J. et al., Students' Use of Animations for Algorithm Understanding, http://www.sigchi.org/chi95/proceedings/shortppr/jwn_bdy.htm.

[4] Binary Search, http://www.cosc.canterbury.ac.nz/mukundan/dsal/BSearch.html

[5] Binary Search Applet, http://math.la.asu.edu/~andrzej/java/BS-applet.html

[6] Linear Search, http://www.cosc.canterbury.ac.nz/mukundan/dsal/LSearch.html

[7] Linear Search Algorithm, http://video.franklin.edu/Franklin/Math/170/common/mod01/linearSearchAlg.html

За контакти:

доц. д-р инж. Цветозар Стефанов Георгиев, Катедра “Компютърни системи и технологии”, Русенски университет, Русе 7017, ул. “Студентска” №8,

e-mail: TGeorgiev@ecs.uni-ruse.bg; тел. (++359 82) 888 711; 888 827

Докладът е рецензиран.