Ubuntu Algorithm Classes

Notes

13.04.2012

Classes will be on 13th of April (Friday)
Server: IRC freenode, #ubuntu-classroom (http://webchat.freenode.net)
Online judge with training tasks is available at http://onlinejudge.xn.pl
The survey link is http://www.surveymonkey.com/s/MQK9SKP
Informations : http://bdfhjk.blog.pl
Spanish version is available at https://docs.google.com/document/d/1sL_vqkmFefiAK8nSKEvNfUnfjrAAB7VF90ZkXF70A5o/edit

1. **Binary search**

Link: http://en.wikipedia.org/wiki/Binary_search_algorithm

Binary search is one of the simplest algorithm. It can help, when the task is to find the first value lower than such chosen, in sorted array, using not more than log2l operations of comparison, where l is the number of elements in the array. Practical usage: databases, tricky adjusting fonts or dimensions, presenting data in specified data period, searching in archives like a phone book, debugging - when you know that a bug is at a particular point during the execution of the program, using this method with placing a check point, you can figure where crash actually happens, finding numerical solutions to an equation

Algorithm:

```
/**
 *   lef is the highest known value for which check(i) = true
 *   rgt is the lowest  known value for which check(i) = false
 */
int lef=-1, rgt=n;
while (lef+1<rgt) {
  int mid=(lef+rgt)/2;
  if (check(mid)) lef=mid; else rgt=mid;
}
                                        //Thanks for Robin Lee for proposing this pseudo
code
```

Alternative:

```
int p := 0 //number of first element)
int k := n-1 //number of last element)
while(p<k) {
        int sr = (p+k)/2;
        if (try(sr) = success) p := sr;
                else k := sr-1;
        if (k-p = 1 && try(p+1) = success) p++;    ///This fragment prevent the infinite loop
                else k--;                          ///In the case of sr = p and try(sr) == fail
}
```

Training tasks: Binary Search

2. **Hashing**

Link: http://en.wikipedia.org/wiki/Hash_function

Hashing methods are used, when You need to compare two large portions of data many times and don't want to compare they char after char, but you want to have a magic function, which returns true if one fragment of data is equal to another fragment. Practical usage: comparing files, images, finding phrase in text, securely storing passwords, identification, addressing using names, errors checking (eg. in files)

Standard and practical way to use hashes:

Let $h(s[i..j]) = (s[i] + s[i+1]*p^1 + s[i+2]*p^2 + ... + s[j]*p^{(j-i)}) \mod q$ where p (eg. 13) is a small prime number and q is a big prime number (eg. 10^9+33)

Preprocessing: $h(i) = (h(i+1)*p + s[i]) \mod q$; $h(n+1) = 0$

Function: $h(s[i..j]) = (h(i) - p^{(j-i+1)} \cdot h(j+1)) \bmod q$
Training tasks: Words, Compare Fragments

3. **Introduction to artificial inteligence**

Link: http://fann.sf.net/fann_en.pdf

At classes, I will present usage of simple artificial neural network. We don't want to implement it yet, but to use a FANN library. ANN may be used in many kinds of problems, especially where don't exist deterministic methods. Some examples: detecting language of text, complex matching data to keywords, making a propositions dependent of user preferences, adjusting computer's opponent strategy in games, finding relationships between the results and data,

Training tasks : Coming soon

Thanks for help in preparing classes: Tomasz Garbus (tasks), Piotr Jarosz (tasks), Mateusz Wojtczak (correction), Kamil Magryta (task tester), Adam Trzaskowski (task tester), Chi Cuil (Spanish version)

For extending this notes, please use comment system.
If You want to ask about something related with algorithms, feel free to start new question ticket at https://launchpad.net/algorithms-headquarter

Author: Marek Bardoński