Search Engines

WS 2009 / 2010

Lecture 9, Thursday January 7th, 2010

(Feedback, Programming Languages, UTF-8)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

UNI FREIBURG

Overview of Today's Lecture

UNI FREIBURG

- Your feedback from Exercise Sheet 8
  - Summary of your feedback
  - My comments
  - Things that will change
- Programming Languages
  - Which one is the best: C, C++, Java, Perl, Python, ...?
  - Of course, it's ...
- Character encoding / UTF-8
  - What is UTF-8
  - Why it is so important for (not only) search engines

PART 1: YOUR FEEDBACK

UNI FREIBURG

- First of all, thank you!
  - vast majority was specific, constructive, and with proper tone
- One slide about each of these points
  - the lecture itself
  - mathematics vs. programming
  - the exercises
  - grading
  - the Wiki
  - the tutorials

Feedback: The Lecture Itself

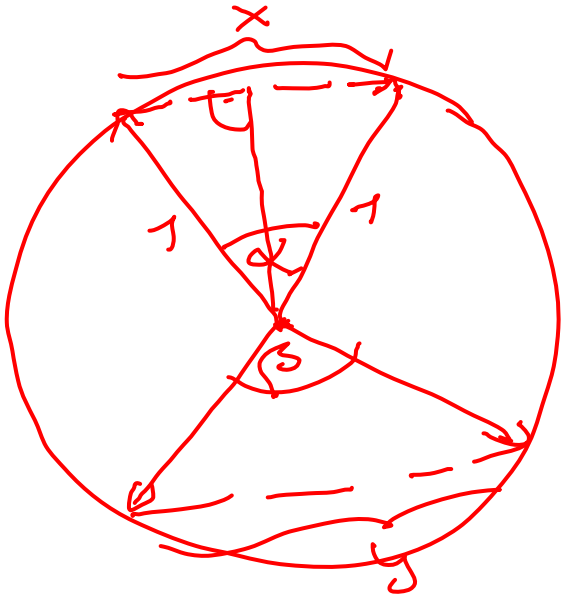
- Mostly positive feedback here
  - most of you like the topic
  - most of you like the atmosphere
  - most of you like the amount, presentation, structure, etc.
  - someone asked for slide numbers
    - I will we happy to add them from now on
  - some people asked for references
    - I will add them from now on, where appropriate
  - some people asked to post slides well before the lecture
    - sorry, you are asking for too much

Feedback: Mathematics vs. Programming

UNI FREIBURG

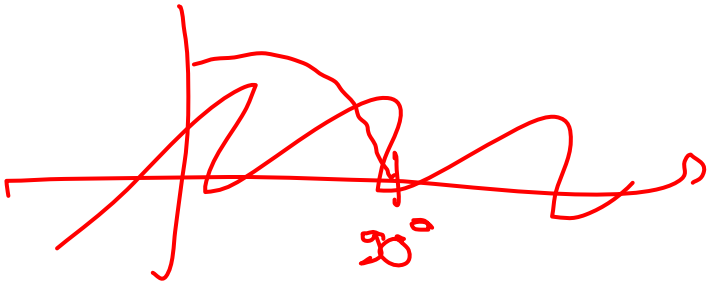
- Most of you like the programming exercises
  - except that they are sometimes too much work

    → see later slide
- Some criticism about the mathematical exercises
  - some said the exercises need special tricks and that's unfair
    - all exercises so far could be solved with elementary maths
  - some wondered about the usefulness of the math exercises
    - I strongly believe that you have to be good at both programming and mathematics
    - without programming you become a Fachidiot
    - without mathematics you don't learn to become precise



$$\cos\frac{\alpha}{2} = \frac{x}{2}$$

$$\cos\frac{\beta}{2} = \frac{y}{2}$$



Feedback: The Exercises

UNI FREIBURG

- Most of you like the exercises themselves, but ...
  - ... most of you find it too much work
  - some also criticized that some exercises require much more work than others, but you get 1 point for each
- We will do the following
  - I agree that an exercise sheet should not take more than 6 hours for a good student
  - that means ~ 1 hour / exercise
  - plus 1 – 2 hours for the writeup / presentation
  - I will try to split tasks such that exercises are of similar complexity

Feedback: Grading

- Most of you don't like the current grading scheme
  - it creates more pressure than motivation
- We will do the following
  - you get a mark for your exercises so far; this will be 25% of your final mark
    - this ensures backwards compatibility
  - the remaining 75% will be computed as follows
    - you will get a separate mark X for the remaining exercises
    - you will get a mark Y for the final exam
    - if Y better than X, take Y, otherwise the average of X and Y
    - I think that is very generous

Grading — Exercises vs. Mid-Term Exam

UNI FREIBURG

- Two grades so far
  - one for the exercises so far
  - one for the mid-term so far (11 people participated)
  - in all cases, the exercise grade was at least as good as the grade in the mid-term exam
  - here are the grades for the exercises (23)

    1.0 x 4, 1.3 x 5, 1.7 x 5, 2.0 x 6, 2.3 x 1, 3.0 x 1, 5.0 x 1
  - here are the grades for the mid-term exam (11)

    1.0, 1.3, 2.0, 2.0, 2.3, 2.3, 2.7, 3.3, 4.0, 5.0, 5.0
- BTW, here is the date of the final exam
  - Friday, March 12, 2010, 14:00h, in HS 026

Feedback: The Wiki

UNI FREIBURG

- Some criticism here
  - some asked why not a forum?
    - well, I don't think the difference is so large, except that it would be more work for us
    - note that you can subscribe to any page, i.p. the front page, then it becomes like a mailing list
  - many of you complained about the edit conflicts when submitting
    - I fully understand, but : why didn't you tell us earlier???
    - Simple solution: we will create the table with your names and links from now on, and you just upload your solutions (which can be done concurrently)

Feedback: Tutorials

UNI FREIBURG

- Some complaints here
  - contents not well synced with the actual problems students had with the exercise sheets
    - we will now shift tutorials by one week!
    - in particular, exercises will have been corrected then
  - many said they would like a master solution
    - ok, we will provide one from now on (gosh, are we nice)
  - more comments in case not all points were obtained
    - Marjan will try to give more comments
  - other comments: let Marjan comment

PART 2: PROGRAMMING LANGUAGES

UNI FREIBURG

- Which programming language is the best?
  - C, C++, Java, Perl, Python, PHP, ... ?
- Obviously depends on the context, but what about
  - which language is most efficient (in run time) ?
  - which language is easiest / fastest to program ?
  - which language gives the most reliable programs ?
- We will look at two studies
  - an article by Lutz Prechelt: An Empirical Comparison of Seven Programming Languages. IEEE Computer 33(10), 2000.
  - an article by Thomas Bruckschlegel: Micro benchmarking C++, C#, and Java. Dr. Dobb's Journal, July 1, 2005.

An Empirical Comparison of Seven PLs

UNI FREIBURG

- Programming languages investigated:
  - C, C++, Java, Perl, Python + two other script languages
  - a non-trivial program had to be written (phone-book task)
  - about 10 different programmers on average per language
- Suprising result
  - the average run-time was very similar for all seven languages
  - differences in median and best run-time were also not that big
  - development time significantly lower for the script languages
  - bottom line: variation in programmer‘s efficiciency matters more than variations in the language‘s efficiency

But, to be fair ...

- ... let's look at a simpler task
  - for (i = 1; i <= n; ++i) { sum = sum * i + 1; }
  - note: the time-comsuming parts of search engine code (list intersection, decompression, etc.) are more similar to this simple task than to Prechelt's phone-book task
- Let's write the code together and time it: $n = 10^9$
  - in C++ : ≈ 1,6 secs
  - in Java: ≈ 1,5 secs
  - in Perl: ≈ 200 secs

Some Rules of Thumb

UNI FREIBURG

- When 50% run-time improvement matter a lot ...
  - ... then C++ is the programming language of choice
- When a factor of 2 in run-time doesn't really hurt ...
  - ... then Java / C# will give you code faster and with less pain
  - if you are a non-expert in C++, you will find coding in Java / C# by a factor of 2 – 3 faster
  - but even if you are an expert in C++, it will be faster, simply because C++ is so full of subtle details and pitfalls
- When a factor of 100 in run-time is insignificant ...
  - ... then use a script language
  - up to a factor 5 faster to produce code here (rapid prototyping)

Programming Languages — References

UNI FREIBURG

- Lutz Prechelt: An Empirical Comparison of Seven Programming Languages. IEEE Computer 33(10):23-29, 2000.

  http://portal.acm.org/citation.cfm?id=621567

- Thomas Bruckschlegel: Micro benchmarking C++, C#, and Java. Dr. Dobb's Journal, July 1, 2005.

  http://www.ddj.com/cpp/184401976

PART 3: UTF-8

UNI FREIBURG

- What is UTF and why do we need it?
  - UTF = Unicode Transformation Format
  - a standard for encoding all the characters of the world
  - extends the long-standing ASCII / ISO-8859-1

    (which can only differentiate between 256 characters)
- How to encode so many different characters?
  - 1 byte is obviously not enough
  - 2 bytes are also not enough (≤ 65,536 different characters)
  - so take 4 bytes per character → this is what UTF-32 does
  - but the size of strings now quadruples compared to ASCII !
  - and so does the time to process these strings ...

UTF-8 — Properties

- UTF-8 is a variable-byte encoding that realizes all of the following
  - ASCII compatible = a string of characters with ASCII codes < 128 is the same in ASCII as in UTF-8
  - frequent special characters (like ä, á, å) need two bytes, only very rare characters (old scripts) need four bytes
  - no need to decode from left to right, can decode starting from anywhere within a string
  - easy to decode / convert to UTF-32

UTF-8 — The Encoding

- Here is the encoding Unicode → UTF-8
  - Case 1: Unicode in [0, 127] = xxxxxxx (7 bits)

    → UTF-8 code is 0xxxxxxx (1 byte)
  - Case 2: Unicode in [128, 2047] = yyyxxxxxxxx (11 bits)

    → UTF-8 code is 110yyyxx 10xxxxxx (2 bytes)
  - Case 3: Unicode in [2048, 65535] = yyyyyyyyxxxxxxx (16 bits)

    → UTF-8 code is 1110yyyy 10yyyyxx 10xxxxxx (3 bytes)
  - Case 4: Unicode in [65536, 2^{21} - 1] = zzzzzyyyyyyyyxxxxxxxx

    → UTF-8 code is 11110zzz 10zzyyyy 10yyyyxx 10xxxxxx
  - Could continue with 5-byte and 6-byte sequences, but UTF-8 stops here, due to RFC 3629

UTF-8 — Some Obervations

- In a multi-byte sequence
  - all bytes are ≥ 128, and vice versa such bytes occur only in multi-byte sequences
  - the number of leading 1s in the first byte of a multi-byte sequence encodes the length of the sequence
  - the concatenation of the remaining bits (except for the 0 that follows the leading 1s) are called the code point
- For every Unicode in $[0, 2^{21} - 1]$
  - there is exactly one UTF-8 sequence
  - but vice versa not all multi-byte sequences are valid UTF-8
  - for example, the 2-byte sequence 11000000 10xxxxxx , why?
  - Exercise: characterize all invalid sequences

UTF-8 — References

- Wikipedia
  - http://en.wikipedia.org/wiki/Unicode
  - http://en.wikipedia.org/wiki/UTF-8
- The Unicode consortium
  - http://www.unicode.org/
  - http://www.unicode.org/versions/Unicode5.2.0/
  - http://www.unicode.org/charts/
- RFC 3629
  - http://tools.ietf.org/html/rfc3629