**CMPE 478**

**Homework 1, Fall 2022**

**Kadir Gökhan Sezer**

**Demet Yayla**

The algorithm we use is a simple version of the algorithm described in detail at this url(http://infolab.stanford.edu/~backrub/google.html). This application, which seems complicated, is not very difficult for people with graph theory knowledge. First of all, if we try to understand the purpose of the algorithm, 1990 and 2000 were the years when the web was born. With the unexpectadly fast growth on its content volume, it was understood that there was a need for search engines. The lack of technological growth (with respect to hardware) required some advance algorithms to emerge on the topic: In 1994, a few professors at Stanford University came up with a ranking algorithm, the very algorithm Google utilized in its foundation phase.

Diagram

Description automatically generated

<https://delante>.co/definitions/pagerank/

The purpose of this algorithm is to determine the order of site suggestions that an ordinary user will come across while browsing the internet with the aim of benefiting the user with the most related content it has to the spesific query. The app simply gives each site a value it has calculated. It then sorts the sites according to this value. This value is directly proportional to the links it receives. In other words, the more links a site gets, the more valuable it becomes according to this algorithm.

In essence, we used a simpler variation of the algorithm. The original formulae is as below:

Text

Description automatically generated

<https://delante.co/definitions/pagerank/>

The simpler version we implemented is as below:

Text, letter

Description automatically generated

<https://www.cmpe.boun.edu.tr/tr/people/can.ozturan>

In the version we use, the algorithm has been simplified and made short-term calculations and analyzes possible.

The pseudocode for the PageRank algorithm.

Text

Description automatically generated

<https://www.ccs.neu.edu/home/daikeshi/notes/PageRank.pdf>

The implementation we used:

Text

Description automatically generated

Details of the machine we used:

Graphical user interface

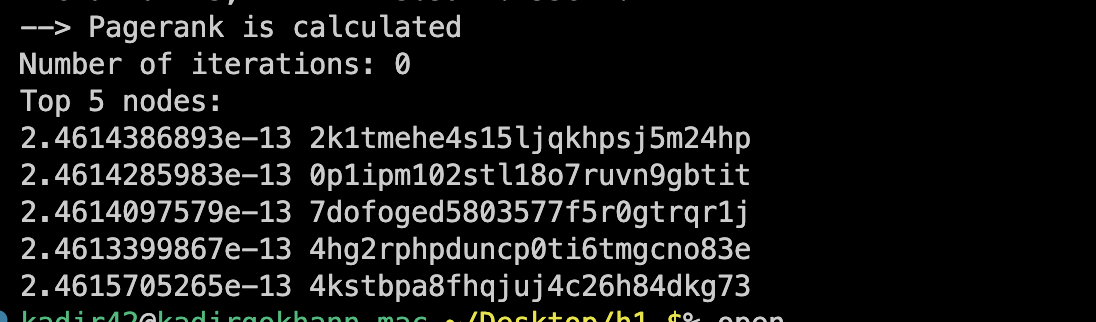
Description automatically generated

Graphical user interface, text

Description automatically generated

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  | | Essentials | | Product Collection,9th Generation Intel® Core™ i7 Processors | | Code Name,Products formerly Coffee Lake | | Vertical Segment,Mobile | | Processor Number,i7-9750H | | Status,Launched | | Launch Date,Q2'19 | | Lithography,14 nm | | Recommended Customer Price,$395.00 | |  | | CPU Specifications | | Total Cores,6 | | Total Threads,12 | | Max Turbo Frequency,4.50 GHz | | Processor Base Frequency,2.60 GHz | | Cache,12 MB Intel® Smart Cache | | Bus Speed,8 GT/s | | TDP,45 W | | Configurable TDP-down,35 W | |  | | Supplemental Information | | Embedded Options Available,No | | Datasheet,View now | |  | | Memory Specifications | | Max Memory Size (dependent on memory type),128 GB | | Memory Types,DDR4-2666 | LPDDR3-2133 | | Max # of Memory Channels,2 | | Max Memory Bandwidth,41.8 GB/s | | ECC Memory Supported ‡,No | |  | | Processor Graphics | | Processor Graphics ‡,Intel® UHD Graphics 630 | | Graphics Base Frequency,350 MHz | | Graphics Max Dynamic Frequency,1.15 GHz | | Graphics Video Max Memory,64 GB | | Graphics Output,eDP/DP/HDMI/DVI | | 4K Support,Yes | at 60Hz | | Max Resolution (HDMI)‡,4096 x 2304@30Hz | | Max Resolution (DP)‡,4096 x 2304@60Hz | | Max Resolution (eDP - Integrated Flat Panel)‡,4096 x 2304@60Hz | | Max Resolution (VGA)‡,N/A | | DirectX\* Support,12 | | OpenGL\* Support,4.5 | | Intel® Quick Sync Video,Yes | | Intel® InTru™ 3D Technology,Yes | | Intel® Clear Video HD Technology,Yes | | Intel® Clear Video Technology,Yes | | # of Displays Supported ‡,3 | | Device ID,0x3E9B | |  | | Expansion Options | | PCI Express Revision,3.0 | | PCI Express Configurations ‡,Up to 1x16 | 2x8 | 1x8+2x4 | | Max # of PCI Express Lanes,16 | |  | | Package Specifications | | Sockets Supported,FCBGA1440 | | Max CPU Configuration,1 | | TJUNCTION,100°C | | Package Size,42mm x 28mm | |  | | Advanced Technologies | | Intel® Optane™ Memory Supported ‡,Yes | | Intel® Speed Shift Technology,Yes | | Intel® Thermal Velocity Boost,No | | Intel® Turbo Boost Technology ‡,2.0 | | Intel® Hyper-Threading Technology ‡,Yes | | Intel® Transactional Synchronization Extensions,No | | Intel® 64 ‡,Yes | | Instruction Set,64-bit | | Instruction Set Extensions,Intel® SSE4.1 | Intel® SSE4.2 | Intel® AVX2 | | Intel® My WiFi Technology,Yes | | Idle States,Yes | | Enhanced Intel SpeedStep® Technology,Yes | | Thermal Monitoring Technologies,Yes | | Intel® Flex Memory Access,Yes | | Intel® Identity Protection Technology ‡,Yes | |  | | Security & Reliability | | Intel® AES New Instructions,Yes | | Secure Key,Yes | | Intel® Software Guard Extensions (Intel® SGX),Yes with Intel® ME | | Intel® Memory Protection Extensions (Intel® MPX),Yes | | Intel® OS Guard,Yes | | Intel® Trusted Execution Technology ‡,No | | Execute Disable Bit ‡,Yes | | Intel® Stable IT Platform Program (SIPP),No | | Intel® Virtualization Technology (VT-x) ‡,Yes | | Intel® Virtualization Technology for Directed I/O (VT-d) ‡,Yes | | Intel® VT-x with Extended Page Tables (EPT) ‡,Yes | |
|  |

<https://www.intel.com.tr/content/www/tr/tr/products/sku/191045/intel-core-i79750h-processor-12m-cache-up-to-4-50-ghz/specifications.html>

Results:Text

Description automatically generated

Discussion of the results:

There is one thing that made me glad I did this project. While following the lesson regularly, we talked about the importance of scheduling. I was a little curious about how these methods were written in the background, in fact, I reached [this link](https://github.com/OpenMP/sources) afterwards. And when I looked, I saw that it actually uses the libraries that we used in the Operating System course. When I reviewed the continuation, I couldn't comprehend much because the code was a bit foreign to me. But I can tell through this experiment that there is not much difference between scheduling methods, except dynamic. Dynamic can really break the system sometimes. Chunk size changes a little bit about static or guided, but when dynamic, chunk really matters. Afterwards, I can say that the number of threads creates different results. There is too much time between 1 thread and 2 threads. But the time difference between 1 and 2 and the time difference between 2 and 3 are also different. The earnings go down which implies diminishing returns in increase of speed as we were expecting from our knowledge from the lectures. In short, we can say that between 1 and 8 there is really more than 50% speed increase. But of course, all of these can be re-interpreted according to the operating system we use, the CPU and the compatibility of the algorithm we write with parallelism.

How to compile and requirements:

This features were tried on MacOS(which can support G++).

**G++-12 -fopenmp main.cpp -o main.out**

g++: G++ is a compiler, not merely a preprocessor. G++ builds object code directly from your C++ program source. There is no intermediate C version of the program. (By contrast, for example, some other implementations use a program that generates a C program from your C++ source.) Avoiding an intermediate C representation of the program means that you get better object code, and better debugging information. The GNU debugger, GDB, works with this information in the object code to give you comprehensive C++ source-level editing capabilities. (<https://gcc.gnu.org/onlinedocs/gcc-3.3.6/gcc/G_002b_002b-and-GCC.html>)

g++-12: We used OpenMP 5.0, so we needed to use G++ version of 12. You can check your version just by executing this command “g++ --version”. If your version below, you may need to update your g++.

-fopenmp: The flag is let you code in paralell computing.

-o main.out: It is just to name the output.