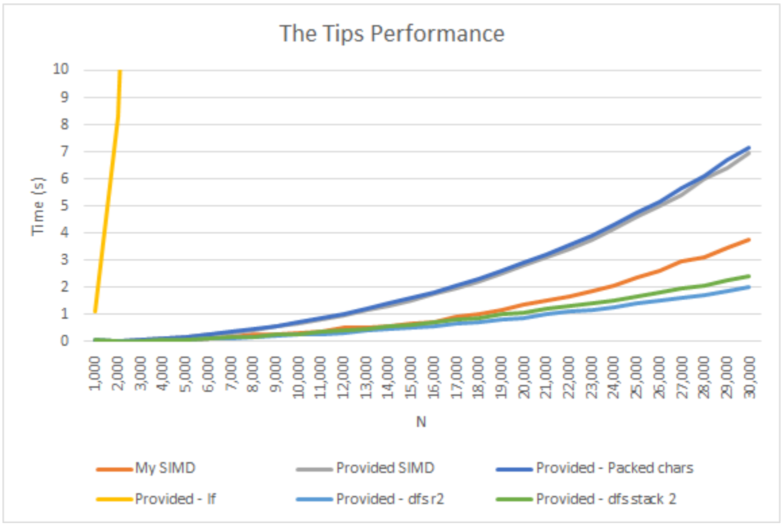
note 24 views

TheTips performance

The lecture notes for TheTips show SIMD greatly outperforming dfs. When I run the executables in ~plank/cs494/notes/Floyd/ on a machine in the EECS Hydra lab dfs is much faster than the SIMD.



I suspect the cache utilization is to blame. For N = 1500 the dfs-r2 executable only misses D1 131,908 times. For the same N, bits-packed-SIMD misses D1 3,248,887 times. For reference, my SIMD implementation misses D1 3,210,055 times but has less than half the memory accesses and the IF implementation misses 55,559,333 times.

lectures lab5

~ An instructor (James S Plank) thinks this is a good note ~

Updated 18 days ago by Gregory Rouleau

followup discussions *for lingering questions and comments*

● Resolved ○ Unresolved

James S Plank 18 days ago
I'll look into it when I have time -- thanks for exploring it and letting me know -- JP

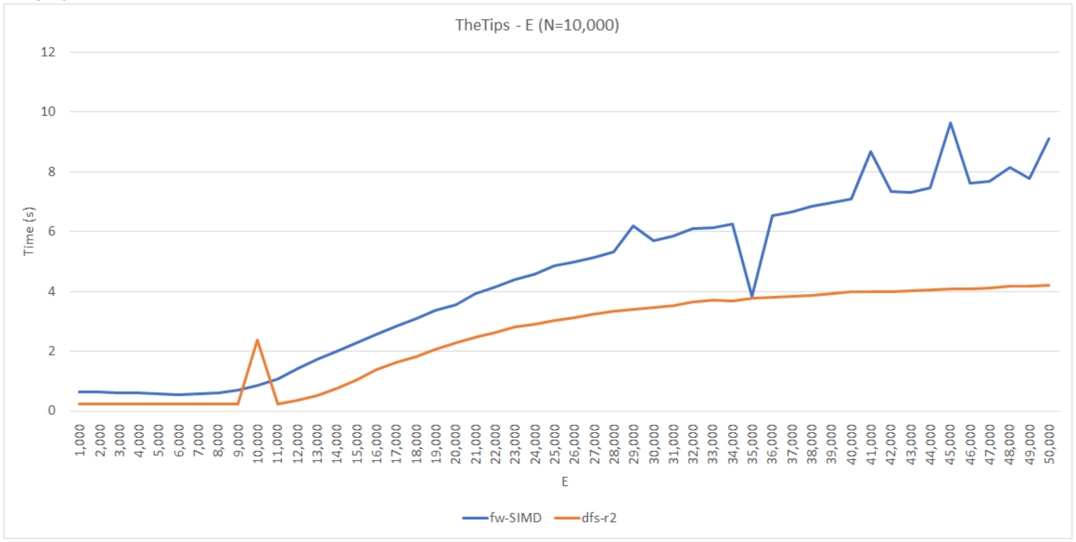
● Resolved ○ Unresolved

YuPei 18 days ago
DFS will have less operation than FW, if the graph is sparse, since it doesn't need to go down that many level of edges. But for FW, we don't care about the edges and just brute force the adjacency matrix for all the vertices.

```
ypei2:hydra8 ~/cs494/labs/Lab-5-SIMD> time ./tip-dfs-r2 8000 20000 20 N
7095.1608922986
./tip-dfs-r2 8000 20000 20 N  1.81s user 0.03s system 99% cpu 1.836 total
ypei2:hydra8 ~/cs494/labs/Lab-5-SIMD> time ./tip-fw-bits-packed-SIMD 8000 20000 20 N
7095.1608922986
./tip-fw-bits-packed-SIMD 8000 20000 20 N  1.67s user 0.03s system 99% cpu 1.704 total
```

So what's your input for number of edges, and I am curious what you mean by half the memory access?

Gregory Rouleau 18 days ago

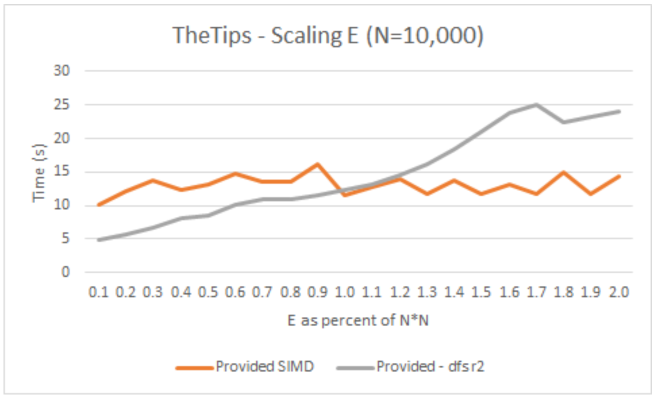


FW does care about the edges. The if statement

```
if (C[i][v])
```

causes the FW algorithm to depend on the density as seen above.

Some quick runs on Hydra with larger values of E:



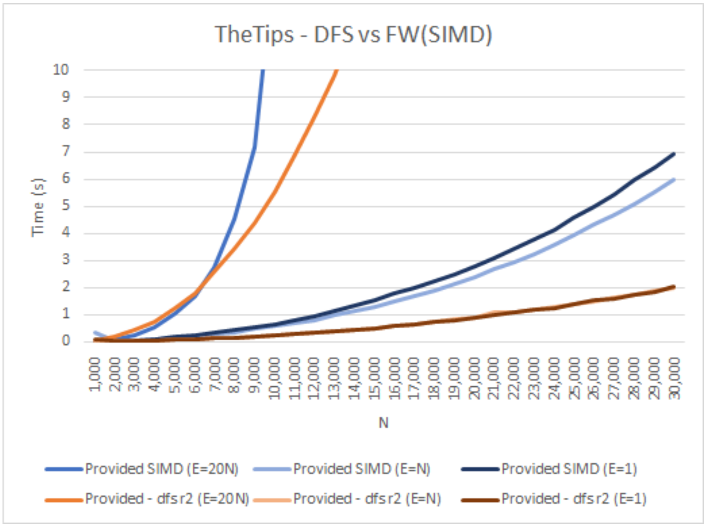
Note the data in these tests is quite noisy because it was run during the day on a Hydra machine.

The above data shows a crossover at about 1%; to put this into the scale of the problem, with N=10,000 and E at 1% of N*N (E=1,000,000) each clue would have an average of 100 locations.

Based on these very preliminary results beyond about 1% density F-W(SIMD) seems to perform better than the DFS but much further testing is required to validate this.

The dfs should perform about $O(N*(N+E))$ because N is the number of vertices and E is the number of edges.
The f-w should perform about $O(N*E+N^2)$ because the inner loop of the Floyd calculation only executes if there is an edge (probability = $E/(N^2)$) and the probability calculation requires $O(N^2)$.

Below is some more data for runs where E depends on N:



I will run more tests later to further test performance when E=x*N*N.

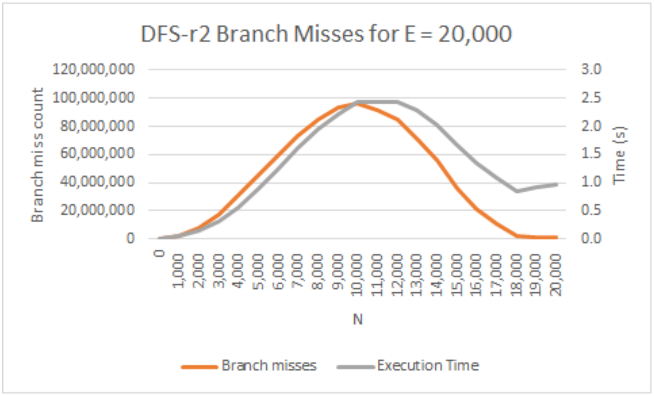
EDIT: fixed dfs runtime (original omitted that dfs must be run for each starting node)

Gregory Rouleau 18 days ago Regarding memory accesses: I used Cachegrind to profile the cache usage. My implementation performed significantly less memory access.

YuPei 17 days ago Yeah the complexity analysis makes sense, I am surprised that FW can even beat DFS in some cases, since SIMD can only lower the constant before the big O. Haven't done it myself yet but thanks for the info regarding cachegrind, should be very useful!

[✓ Resolved] Unresolved

Gregory Rouleau 16 days ago
In case anyone else has too much free time and decides to time with a variety of inputs and notices that the execution time is not monotonically non-decreasing, you might be intrigued to know that branch misses may be to blame:


DFS-r2 Branch Misses for E = 20,000
Branch miss count
Time (s)
N
Branch misses
Execution Time