CS280 – Algo Analysis and B-Lists
January 20, 2016

http://azrael.digipen.edu/~mmead/www/Courses/CS280/Analysis.html

Misc. Things about Memory Manager

- Microsoft gives warnings about "strcopy being unsafe" and other dumb stuff. "These warnings are for people who don't know what they are doing." We DO know what we are doing, so ignore these dumb Microsoft only warnings
- We are mostly looking for memory leaks and over/under flows when running Dr. Memory
- If we are going to use the memory manager in our game, we need to DEBUG IT HEAVILY beforehand.

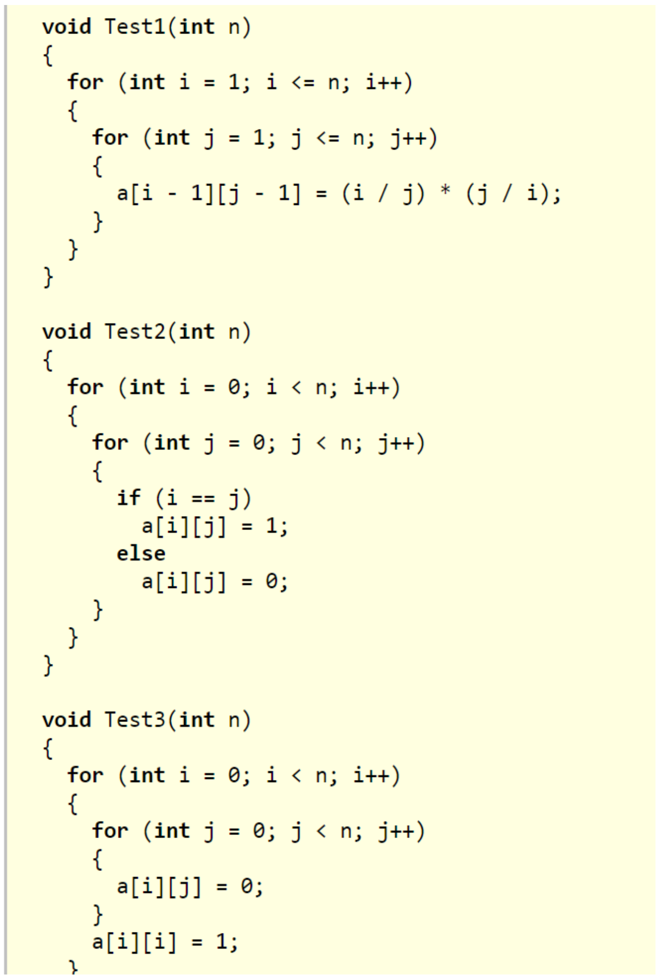
Misc. Things

- Coverage tools show what code gets executed
  - Must write test cases for every path through code
- Profiler diagram on website
  - The bold lines are our code, other lines are child function calls
- Use profilers to ssee where you should spend time working on performance (don't waste time microoptimizing!)
- Never print during profiling, it is extremely slow!
- Cache Grind, kCacheGrind, Visual Studio (profilers)
  - We need to get familiar with profilers to be efficient programmers
- Memory alllocation/de-allocation is very expensive
- In a real-world situation, you would run one test at a time, reboot, then net one. Tests can start affecting each other, otherwise
- Conditionals can <u>extremely</u> slow down code because the CPU can't predict the outcome

- Sometimes it is much more efficient to do some 'extra work', such as just overwriting some data twice instead of doing a check before each write

- If you wanted to loop through an arra twice, you would loop through forwards, then backwards for cache consistency
  - This is an example o a micro-optimiation, you would only spend time on something like this if you ran a profiler and found this was a substantial part of your execution time

Algorithm Analysis



```
void Test1(int n)
{
  for (int i = 1; i <= n; i++)
  {
    for (int j = 1; j <= n; j++)
    {
      a[i - 1][j - 1] = (i / j) * (j / i);
    }
  }
}

void Test2(int n)
{
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      if (i == j)
        a[i][j] = 1;
      else
        a[i][j] = 0;
    }
  }
}

void Test3(int n)
{
  for (int i = 0; i < n; i++)
  {
    for (int j = 0; j < n; j++)
    {
      a[i][j] = 0;
    }
    a[i][i] = 1;
  }
```

How to analyze this diagram:

Operation	Loop1	Loop2	Loop3
Comparisons	0	N^2	0
Assignments	N^2	N^2	N^2 + N
Multiplications	N^2	0	0
Divisions	2N^2	0	0
Total	4N^2	2N^2	N^2+N
BigO	N^2	N^2	N^2

They all have the same BigO! On paper, they are similar. Look at stress tests on website to see the loops in action.

http://azrael.digipen.edu/~mmead/www/Courses/CS280/BigO-CodeExamples.html

Explaination of this diagram

- Case 5 – 2N
- Case 6 – N^2
- Case 7 – N^2
- Case 8 – N^2
- Case 9 – 10N^2
- Case 10 – N^3
- Case 12 – O(N) -- **Ω** (C) – Avg: Can't tell
- Case 13 – O(N) -- **Ω** (0) – Avg: Can't tell
    - *You can't tell the average in the two above because they have conditionals*
- Case 14 – Log_2N

- Case 15 – $\log_3 N$
- Algorithm Analysis – Things must remain in their original state after the algorithm is complete
  - Sorted stuff has to stay sorted
  - Arrays must remain contiguous
  - Etc...

B-List Assignment

- Multiple items per node
- Use a binary search on the nodes (keep the nodes sorted)
- Templatized for type and size
  - <int, 16>
- Don't dynamically allocate the array inside of the node
  - We will have a fixed size
- Do the insert function last, it is the hardest function
- The hardest assignment in the class, start early
  - Do simple stuff first!
- There is a point where increasing the array size in the B-List makes it start slowing down instead of speeding up. Related to cache!