Bilkent University

CS 473/573 - Algorithms I

HW1 - Kernighan-Lin Algorithm

Ege Turan – 21502441

Explanation of Implementation:

I have used linkedlist to present graph vertex and edges. GbtainGraphFromInputs method creates graph struct for the usage of simpler_kl. I have also implemented 2 heaps and their heapify, build_heap and extract_max opearation. Some methods are used to divide algorithm. Like “make_decision” method. I have used this method to control that bisection of the graph improves or not otherwise to backup the graph. I think my runtime is a little bit problematic. My algorithm works for Erdos matrix. But it takes lots time for execution. But I think I implemented the main algorithm properly. Extra Build Heap operation creates this time loss.

Runtime Simpler KL:

First Part

“**repeat**: Each repeat loop corresponds to a pass”.

n is vertex count and k are edge count.

n/2 * O (n + k) = O (n^2 + k) pass in total for each iteration. And put D values into the HEAPs.

If k is equal to node count in worst case algorithm works O(n^3). But in average edge count is less than vertex count.

Second Part:

For each pass I have used one build heap and extract max operation for finding max D values for partition A and partition B.

O(n/2) pass count, O(n) is Build Heap, and O(logn) Extract Max.

= O(n^2logn) in total for taking max D values at each iteration.

Swap is handled by the pointers thus it takes O (1) time.

Third Part:

Inside the MakeDecision method. It looks at the gain values in the iteration and if the g is less than O(n) then remove

In General:

Worst case runtime is: O(n^3) if the number of edges is equal to the number of vertices.

2. I have used Networkx library’s KL implementation. It was faster in the Erdos Matrix but I couldn’t test it on

```
t = nx.algorithms.community.kernighan_lin.kernighan_lin_bisection(G, partition=None)
```

But for com-DBLP and rgg n 2 20 s0 I have taken the following error.

```
MemoryError: Unable to allocate 749. GiB for an array with shape (317080, 317080) and data
type float64
```

Thus, I couldn’t test for them.

3. GNU profiler

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 74.03      4.03     4.03        1     4.03     5.45  simpler_kl_a
 24.16      5.35     1.32 54078238     0.00     0.00  heapify
  1.84      5.45     0.10    62354     0.00     0.00  build_heap
  0.09      5.46     0.01                             frame_dummy
  0.00      5.46     0.00    62354     0.00     0.00  extract_max
  0.00      5.46     0.00    31177     0.00     0.00  swapNodes
  0.00      5.46     0.00    29592     0.00     0.00  swapNodes2
  0.00      5.46     0.00       10     0.00     0.00  backup_graph
  0.00      5.46     0.00        9     0.00     0.00  makeDecision
  0.00      5.46     0.00        1     0.00     0.00  obtainGraphFromInputs
  0.00      5.46     0.00        1     0.00     0.00  removeGraph

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.
```

I think I made lots of non-necessary calls to the build heap and this took lots of time. But I couldn't solve this problem.