Khoa Diep
CSCE 313 - 501
2/7/20

Analysis:

My code worked for Ackerman (x, y), where x <= 3 and y <= 8 when 128 MB is allocated. Anything less than 128 MB, the program still runs, but the output of the Ackerman function is wrong. When x is equal to 1 or 2, the differences between the performance of Ackerman(x, y) is seemingly linear. However, when x is equal to 3, the performance of the differing Ackerman(3, y) increase exponentially.

System:

- Intel Core i7, 7th gen
- 16 GB RAM
- OS - Ubuntu and Windows(to write on)

Bottlenecking:

One of the bottlenecks that came into mind, is that I implemented a binary search to remove from the LinkedList, thus removing from a LinkedList has a time complexity of O(n) in the average case scenario. Perhaps a more efficient search algorithm to remove, such as QuickSearch, can be used to find the BlockHeader* to be removed. Or perhaps a hash table can be implemented to find the BlockHeader*.

Data:
This data was recorded on Ubuntu.

Runtime analysis of Ackerman(x, y) (musec)

		y							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
x	**1**	68	172	474	422	436	2246	1599	3048
	2	1750	8544	6406	7606	7343	9161	13126	15550
	3	7838	44124	179619	797774	3177045	12694379	53166255	233128585

Number of allocate/free cycles of Ackerman(x, y)

		y							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
x	**1**	4	6	8	10	12	14	16	18
	2	14	27	44	65	90	119	152	189
	3	106	541	2432	10307	42438	172233	693934	2785999