ProgSet 3

CS 124: Data Structures and Algorithms

Due: Thursday, April 18, 2024 **Denny Cao and Ossimi Ziv**

---

§1 Number Partition

Input	: A sequence of n numbers $A = \{a_1, a_2, \ldots, a_n\}$
Output	: A sequence of n numbers $S = \{s_1, s_2, \ldots, s_n\}$ of signs $s_i \in \{+1, -1\}$ such that the residual sum of the numbers in A is minimized.

Computational Problem: Number Partition

Claim 1.1 — `Number Partition` can be solved in pseudo-polynomial time.

Proof. Suppose the sequence of terms in A sum up to some number b. Then each of the numbers in A has at most $\log b$ bits. We will show there exists a dynamic programming algorithm that solves the `Number Partition` problem that takes time polynomial nb:

- **Subproblems:** Let $D[i, j]$ be whether it is possible for $A[0, i]$ to sum up to j.
- **Recurrence:** The recurrence relation is given by:

$$D[i, j] = (D[i-1, j + a_i] \vee D[i-1, |j - a_i|])$$

  Our recurrence is correct because if $D[i-1, j+a_i]$ is true, then we can subtract a_i from $j + a_i$ (which is obtainable with the first $i-1$ elements) to get j. Similarly, if $D[i-1, |j-a_i|]$ is true, then either we can add a_i to the $j - a_i$ (which is obtainable with the first $i-1$ elements) to get j or subtract $a_i - j$ (which is obtainable with the first $i-1$ elements) from a_i to get j. In doing so, we obtain every possible sum of the first i elements of A.
- **Topological Order:** We solve the subproblems in increasing order of i and j.
- **Base Case:** $D[0, 0] =$ `True` and $D[i, j] =$ `False` for all other i, j.
- **Original:** The original problem is to find the smallest j such that $D[n, j]$ is true, or:

$$\min\{j : D[n, j] == \texttt{True}, j \in [b]\}$$

- **Time Complexity:** The time complexity of this algorithm is $O(nb)$. This is because for each subproblem we check if b sums are possible (The maximum sum is b). When checking if a sum is possible, we take $O(1)$ time to check 2 previous subproblems. Thus, the total time complexity is $O(nb)$ to fill the table. Iterating to find the smallest j such that $D[n, j]$ is true takes $O(b)$ time. Therefore, the total time complexity is $O(nb) + O(b) = O(nb)$.

Therefore, the `Number Partition` problem can be solved in pseudo-polynomial time. □

§2 Karmarkar-Karp

Claim 2.1 — `Karmarkar-Karp` can be implemented in $O(n \log n)$ time

Proof. The algorithm suggests that we are given a list of numbers, A, we select the two largest elements, a_i, and a_j, difference them, replace the larger of the two by the absolute value of their difference, and replace the smaller with 0. Repeat this until there is only one number left. To analyze the time complexity of a potential implementation, we can split the problem into steps:

- **Sorting**: To make it convenient to find the two largest elements, we can create a `max-heap`. Building this structure will take $O(n)$ time given that A has n elements.
- **Selecting**: Actually extracting the two largest elements involves running `extract-max` twice from our max-heap. Each of these extractions and restructuring of the heap afterwards will take $O(\log n)$ time.
- **Comparison**: Comparing the two values and computing their difference involves basic arithmetic and can be done in constant time given the problem definition.
- **Inserting**: Inserting both the absolute value of the difference back into the heap takes $O(\log n)$ time.

Every time we replace one of the elements with zeros, we are one step closer to the algorithm terminating, going from n steps left to 1 step left. (given that it ends when the two max's are a number and zero). So the number of iterations of the [select, compare, insert loop] that we have to do is $n - 1$. As a result, our time complexity is $O(n * (\log n + 1 + 1)) + O(n) = O(n \log n + n) + O(n) = O(n \log n)$.

Therefore we have shown it is possible to implement the Karmarkar-Karp algorithm in $O(n \log n)$. □

§3 Using Karmarkar-Karp as a starting point

Using KK as a starting point for the various random algorithms we get varying levels of improvement depending on the algorithm.

- **Repeated Random**: using the starting point from KK in the repeated random algorithm will not necessarily have a big impact on its efficiency. Given that the algorithm randomly generates solutions anyway, any improvement will be a good improvement over the already good KK partition, but the random nature of the algorithm's selection would be equally likely to arrive to that bound anyway, so while the worst case scenario return is better from KK, there would be no other substantial improvement.
- **Hill Climbing**: This case is relatively similar to the previous. Setting KK as our starting point in Hill Climbing will definitely allow us to refine the solution further per the nature of the hill climbing algorithm selecting the best possible neighbor. That said, it will potentially trap us in a local minimum residue that isn't the global minimum. Even still, already having a strong starting point will provide a good jumping off point and any improvement will be even closer to ideal. Though when max-iter is very large, it may arrive at a similar, if not better solution regardless of starting point by continuously improving.

- **Simulated Annealing**: KK will provide the most significant improvement here. Given the already strong solution from KK, the exploration involved in not picking an always better neighbor will allow for ideal use of this jumping off point. The original random starting point has a good chance to be too far away to explore ideal possibilities fully before temperature allows exploration. With KK's near-optimal residue, conditions are perfectly suited for the simulated annealing to explore neighboring solutions and quickly approach the minimum residue.

- **Effect on Prepartition**: Using KK to start for the prepartition will create a binary assignment to two sign groups among which the elements will be split. This will create a P array such that the numbers are all 1 or 2. Consequently, the selection of neighbors will vary than if we had pre-partitioned normally where the group assignment is variable $1...N$. The swaps would be between two selected sets of elements. This would put us closer to an ideal solution by the nature of the strong partitioning obtained by KK, making neighbor selection more efficient for pre-partitioned random algorithms.

§4 Experimental Data

Trial	KK	Repeated Random	Hill Climbing	Simulated Annealing	Repeated Random PP	Hill Climbing PP	Simulated Annealing PP
1	1.5269E+04	9.2297E+07	3.2599E+08	3.4130E+08	8.9000E+01	4.5000E+01	8.2100E+02
2	7.7338E+04	8.4121E+07	6.6826E+08	2.7678E+12	2.9600E+02	1.0120E+03	3.2600E+02
3	1.1203E+05	2.4554E+08	2.7211E+08	2.3554E+12	6.0000E+00	1.5460E+03	2.0000E+00
4	8.1003E+04	8.4224E+07	2.4709E+08	1.2565E+08	3.5000E+01	1.2500E+02	6.1100E+02
5	3.7228E+05	2.6170E+07	1.0476E+09	2.4529E+12	3.8000E+02	3.3280E+03	4.2200E+02
6	1.2562E+06	3.4631E+07	1.1780E+08	1.6020E+12	1.5200E+02	4.3600E+02	1.4380E+03
7	8.5595E+05	1.1392E+08	6.8701E+08	3.5500E+12	1.6400E+02	2.1940E+03	2.6000E+01
8	1.1060E+04	4.9036E+08	2.4033E+08	6.7468E+07	1.4800E+02	2.9600E+02	4.2000E+01
9	4.7816E+04	1.4039E+08	3.6860E+08	1.8312E+12	2.5000E+02	3.3200E+02	2.1200E+02
10	3.3936E+05	4.0383E+07	4.2453E+08	2.6042E+11	7.5000E+01	1.5290E+03	6.5000E+01
11	8.9918E+04	3.1190E+08	3.3440E+08	7.1082E+11	1.1400E+02	1.9600E+02	5.2600E+02
12	1.0189E+05	3.2844E+08	2.5067E+08	1.5048E+08	2.2000E+01	3.1000E+02	4.5000E+02
13	1.8042E+05	7.5586E+07	8.3096E+07	1.0534E+09	3.0100E+02	8.4700E+02	8.6300E+02
14	7.0068E+05	1.9517E+09	3.9282E+07	6.8730E+08	3.1000E+02	1.1800E+02	4.8000E+02
15	4.0577E+05	8.9349E+07	1.2883E+08	5.1888E+08	2.4300E+02	9.9000E+01	1.0000E+00
16	6.0978E+05	3.7563E+08	2.0401E+08	1.8722E+08	6.0000E+01	2.2920E+03	8.9600E+02
17	3.9164E+04	2.5967E+08	3.6654E+08	4.7549E+11	5.8000E+01	7.0400E+02	4.2000E+02
18	1.0526E+04	2.7780E+08	3.9465E+08	2.0562E+11	8.1600E+02	2.3200E+02	3.1660E+03
19	3.5846E+05	2.5925E+08	6.8376E+08	3.8172E+07	5.5900E+02	4.1700E+02	8.3000E+01
20	8.1528E+04	2.2330E+08	4.6781E+08	1.3321E+12	1.6400E+02	8.6000E+01	3.8000E+01
21	1.2118E+05	1.0663E+08	8.8258E+08	1.2051E+11	6.5300E+02	5.3000E+01	1.1000E+01
22	4.4672E+04	2.7321E+08	3.9867E+08	1.1607E+08	3.4600E+02	8.8000E+02	1.2000E+01
23	4.9547E+04	2.6060E+08	5.9217E+07	1.9418E+12	1.3300E+02	6.6100E+02	3.7100E+02
24	3.7817E+04	2.4918E+08	1.7899E+08	1.5786E+12	3.1000E+01	9.7100E+02	3.9900E+02
25	1.5693E+05	2.6975E+08	1.3960E+08	7.2783E+07	3.6800E+02	5.5800E+02	5.3400E+02
26	3.7676E+05	1.3048E+08	2.7662E+07	8.1614E+10	3.9000E+01	1.7500E+02	6.9000E+01
27	8.7432E+04	2.2009E+08	1.1457E+09	7.6135E+11	2.2000E+01	3.7200E+02	4.2000E+02
28	9.5367E+04	9.6528E+07	7.5871E+07	5.4164E+08	5.5000E+01	1.3770E+03	2.0900E+02
29	9.5827E+05	1.0668E+08	1.4818E+08	1.5887E+08	3.8700E+02	2.3700E+02	2.1700E+02
30	2.5586E+05	5.1707E+07	5.2706E+07	4.0350E+11	1.0200E+02	1.9800E+02	1.1520E+03
31	2.0241E+04	2.0275E+08	2.5450E+08	1.2058E+12	1.5100E+02	1.1150E+03	9.5300E+02
32	8.7144E+04	6.0865E+07	3.8646E+08	7.4584E+07	7.8000E+01	2.6800E+02	3.5800E+02
33	3.7411E+04	2.1028E+08	1.9882E+08	6.0480E+06	1.0900E+02	8.8100E+02	5.9000E+01
34	4.5029E+05	4.8641E+08	5.8740E+07	1.1530E+12	3.6300E+02	9.6900E+02	1.1000E+01
35	5.1998E+05	2.7304E+08	7.7025E+07	1.4598E+12	1.1900E+02	6.8700E+02	8.0100E+02
36	1.0024E+06	3.2833E+08	5.3115E+08	6.6771E+11	2.9100E+02	3.8500E+02	4.2900E+02
37	2.6817E+04	5.1811E+08	1.5846E+08	2.6764E+12	1.1300E+02	2.9300E+02	4.0700E+02
38	6.0440E+03	3.5675E+08	2.0014E+08	4.6254E+07	2.8000E+01	4.2000E+01	3.5800E+02
39	1.2156E+04	9.3280E+07	1.7574E+08	6.1372E+11	1.5400E+02	4.7800E+02	5.3800E+02
40	2.0252E+05	5.5198E+08	1.1352E+08	2.0969E+08	4.6000E+01	7.2000E+02	2.3400E+02
41	2.1161E+04	4.4641E+08	1.6415E+08	4.7513E+07	2.7000E+01	2.4090E+03	2.4700E+02
42	4.4796E+05	3.2885E+08	2.1893E+07	6.6456E+11	9.8000E+01	8.4000E+01	1.8000E+01
43	3.3886E+05	3.0874E+08	1.4521E+08	1.2108E+09	2.7700E+02	6.7000E+01	5.1500E+02
44	1.9766E+04	5.5969E+07	1.9389E+07	4.3879E+08	6.8000E+01	1.3080E+03	2.5600E+02
45	2.2385E+05	3.3939E+07	9.9334E+07	2.0046E+08	1.0000E+00	1.0870E+03	5.9100E+02
46	1.3878E+05	1.5592E+08	1.8094E+09	6.3943E+11	3.1000E+02	9.5000E+02	2.2400E+02
47	4.0279E+05	8.2564E+07	3.2429E+08	1.8769E+12	7.8100E+02	5.6100E+02	1.4900E+02
48	1.4166E+05	6.8653E+07	5.9801E+08	7.5487E+08	4.8300E+02	1.4850E+03	9.4500E+02
49	3.3202E+05	6.5892E+07	4.6061E+08	1.6803E+12	2.2000E+01	1.6000E+02	2.1824E+06
50	5.1903E+05	6.7774E+08	8.0286E+06	6.9003E+11	3.8000E+01	4.2200E+02	1.2000E+02
Mean	257621.90	251520149.82	325328153.10	715316467373.46	198.70	719.94	44078.50
Min	6044	26169742	8028642	6048041	1	42	1
Max	1256180	1951715984	1809400408	3.54997E+12	816	3328	2182430
Std Deviation	291141.2392	290269734.5	337141299.8	9.29982E+11	196.276487	706.210971	308580.599
25%	45458	85505176	114588014.5	202767981.8	55.75	206.5	92.25
50%	129982.5	215186258.5	222170566	2.33019E+11	126	457	364.5
75%	375638.25	311114235.8	397662923	1.30055E+12	299.75	970.5	537

Table 1: Experimental Data

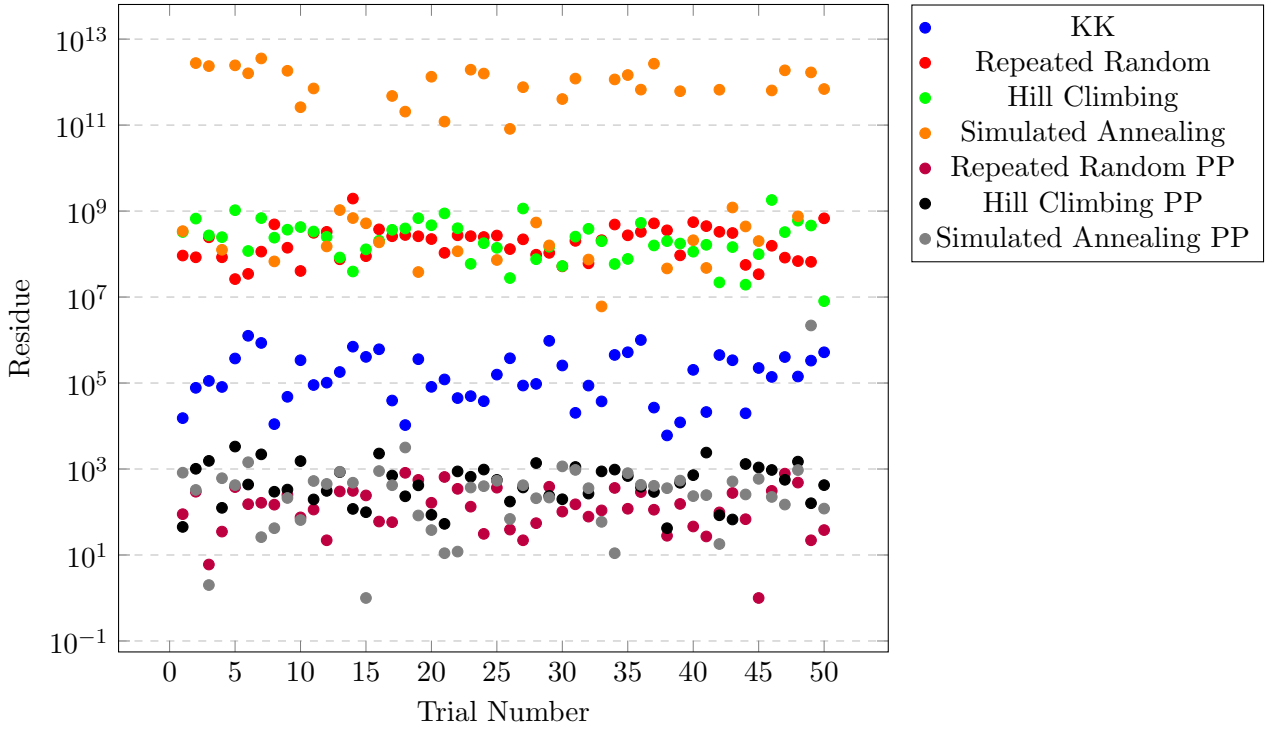

Figure 1: Scatter Plot of Residue for Each Trial

Trial	KK	Repeated Random	Hill Climbing	Simulated Annealing	Repeated Random PP	Hill Climbing PP	Simulated Annealing PP
1	0.0000498	0.1737459	0.0762839	0.1327491	2.2695780	2.3021801	3.3544490
2	0.0000403	0.1724927	0.0762072	0.1291368	2.3137960	2.2517900	3.3835809
3	0.0000398	0.1723869	0.0756602	0.1251338	2.2508330	2.2977090	3.7537591
4	0.0000370	0.1713462	0.0762548	0.3394859	2.3194182	2.1841419	3.4147062
5	0.0000410	0.1707880	0.0754921	0.1272738	2.3502657	2.0690408	3.3559961
6	0.0000370	0.1709871	0.0757141	0.1278837	2.2917082	2.3802140	3.7146499
7	0.0000451	0.1744101	0.0763907	0.1317291	2.2785227	2.1749268	3.3309193
8	0.0000381	0.1742330	0.0751688	0.1267581	2.3100679	2.1675520	3.3528168
9	0.0000401	0.1737361	0.0758038	0.1280441	2.2893219	2.3284140	3.7099099
10	0.0000372	0.1704578	0.0754380	0.1287229	2.3071690	2.1330559	3.3797269
11	0.0000463	0.1706779	0.0797141	0.1313732	2.4651980	2.1813049	3.4513681
12	0.0000367	0.1712940	0.0760572	0.1271760	2.3081019	2.5231960	3.7397938
13	0.0000401	0.1699619	0.0763922	0.1270730	2.2932410	2.1812778	3.3924522
14	0.0000379	0.1712461	0.0782928	0.1276569	2.3272240	2.3807909	3.4191968
15	0.0000370	0.1708589	0.0760951	0.1268878	2.2610803	2.3883710	3.4081421
16	0.0000412	0.1719148	0.0766768	0.1274631	2.3230910	2.2513607	3.6493833
17	0.0000370	0.1708090	0.0772660	0.1276259	2.2906580	2.2259090	3.4344270
18	0.0000379	0.1715560	0.0767398	0.1298370	2.4666100	2.1525152	3.3342280
19	0.0000379	0.1725812	0.0773578	0.1258802	2.3071301	2.3149028	3.7199891
20	0.0000379	0.1707163	0.0752959	0.1292858	2.3845742	2.3809278	3.4587970
21	0.0000372	0.1751258	0.0772760	0.1310220	2.3407371	2.2174931	3.4465122
22	0.0000420	0.1744070	0.0774169	0.1257160	2.3200600	2.4751730	3.7569017
23	0.0000391	0.1794851	0.0777628	0.1312799	2.3635783	2.2402368	3.4789140
24	0.0000370	0.1738722	0.0773497	0.1314101	2.3791220	2.2072430	3.4395320
25	0.0000389	0.1728961	0.0779798	0.1324582	2.4974160	2.5075061	3.3959639
26	0.0000379	0.1760552	0.0778890	0.1304920	2.2951190	2.3029761	3.4496539
27	0.0000379	0.1727281	0.0778530	0.1300452	2.3870811	2.2602928	3.5214531
28	0.0000372	0.1755679	0.0778639	0.1303890	2.3617990	2.1910889	3.4843800
29	0.0000370	0.1734970	0.0797510	0.1281152	2.3374379	2.2319980	3.5128970
30	0.0000460	0.1747911	0.0761158	0.1317821	2.3712950	2.4243093	3.4475751
31	0.0000410	0.1759791	0.0784571	0.1330299	2.3680520	2.3309002	3.4857910
32	0.0000458	0.1819363	0.0765138	0.1274421	2.5712850	2.2515202	3.6413062
33	0.0000403	0.1765049	0.0779109	0.1315830	2.3403220	2.3693709	3.7064593
34	0.0000381	0.1810451	0.0782709	0.1296499	2.3574369	2.2247143	3.4063742
35	0.0000379	0.1739922	0.0771210	0.1330831	2.3393092	2.3797891	3.4775290
36	0.0000401	0.1743562	0.0775728	0.1305189	2.3428280	2.3228621	3.6035342
37	0.0000403	0.1760478	0.0778461	0.1329112	2.4111078	2.2998772	3.5291059
38	0.0000372	0.1739519	0.0770049	0.1280622	2.3277388	2.2556989	3.4497991
39	0.0000379	0.1750472	0.0778410	0.1318719	2.5900331	2.3984060	3.6570110
40	0.0000458	0.1749792	0.0769870	0.1278980	2.3608530	2.3829482	3.4980040
41	0.0000379	0.1753640	0.0781410	0.1292958	2.3391311	2.2218969	3.4909050
42	0.0000389	0.1746020	0.0777249	0.1316762	2.3725479	2.3424420	3.6656001
43	0.0000381	0.1750181	0.0779228	0.1311851	2.3418598	2.1885428	3.4631932
44	0.0000472	0.1759679	0.0770171	0.1333389	2.4145181	2.1256559	3.4564631
45	0.0000370	0.1739819	0.0774729	0.1280079	2.3531032	2.3489411	3.6098731
46	0.0000391	0.1738150	0.0769639	0.1332021	2.5430059	2.2056830	3.5488560
47	0.0000381	0.1758399	0.0782881	0.1320779	2.3343511	2.1441040	3.4894700
48	0.0000391	0.1748581	0.0773289	0.1278501	2.4729629	2.2611771	3.4565830
49	0.0000379	0.1761172	0.0772018	0.1330249	2.3664680	2.4703460	1.9093492
50	0.0000379	0.1750410	0.0768602	0.1315291	2.3481619	2.2413547	3.4710712
Mean	0.0000396	0.1739814	0.0771201	0.1339425	2.3591262	2.2818826	3.4741671

Table 2: Time Data for Each Trial

We notice that there are 3 distinct "areas" created by the different algorithms defined by whether they used standard or prepartitioned solutions with the latter producing the best results and the former producing the worst, bounding the middle ground of Karman-Karp.