

MM 117 RotatingNumbers

hoshi524

問題

<https://www.topcoder.com/challenges/30122730>

<https://twitter.com/wleite/status/1253131089775865860>

(gif を貼るとすごい重いので URL を)

2	15	12	3	N: 4 P: 4 Moves: 2 Move Penalty: 3 Loc Penalty: 88 SCORE: 91
1	7	11	4	
13	10	6	8	
9	14	5	16	

Penalty 最小化

- Move Penalty = $\sum (S - 1)^{1.5}$

S = 回転する大きさ

- Loc Penalty = $\sum \text{abs}(gr - r) + \text{abs}(gc - c)$

回転系問題

https://atcoder.jp/contests/rco-contest-2019-final/tasks/rco_contest_2019_final_b

まわしてそろえる

今回の方がひねってない普通の問題
Move Penalty は変わってる

初期状態	操作 (0 0 3) 後	操作 (0 1 3) 後	操作 (1 1 2) 後	操作 (0 0 4) 後																																																																																
<table><tr><td>3</td><td>2</td><td>3</td><td>2</td></tr><tr><td>3</td><td>3</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>2</td><td>2</td></tr></table>	3	2	3	2	3	3	1	0	1	1	0	1	0	0	2	2	<table><tr><td>1</td><td>3</td><td>3</td><td>2</td></tr><tr><td>1</td><td>3</td><td>2</td><td>0</td></tr><tr><td>0</td><td>1</td><td>3</td><td>1</td></tr><tr><td>0</td><td>0</td><td>2</td><td>2</td></tr></table>	1	3	3	2	1	3	2	0	0	1	3	1	0	0	2	2	<table><tr><td>1</td><td>1</td><td>3</td><td>3</td></tr><tr><td>1</td><td>3</td><td>2</td><td>3</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>0</td><td>2</td><td>2</td></tr></table>	1	1	3	3	1	3	2	3	0	1	0	2	0	0	2	2	<table><tr><td>1</td><td>1</td><td>3</td><td>3</td></tr><tr><td>1</td><td>1</td><td>3</td><td>3</td></tr><tr><td>0</td><td>0</td><td>2</td><td>2</td></tr><tr><td>0</td><td>0</td><td>2</td><td>2</td></tr></table>	1	1	3	3	1	1	3	3	0	0	2	2	0	0	2	2	<table><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td><td>3</td><td>3</td></tr><tr><td>2</td><td>2</td><td>3</td><td>3</td></tr></table>	0	0	1	1	0	0	1	1	2	2	3	3	2	2	3	3
3	2	3	2																																																																																	
3	3	1	0																																																																																	
1	1	0	1																																																																																	
0	0	2	2																																																																																	
1	3	3	2																																																																																	
1	3	2	0																																																																																	
0	1	3	1																																																																																	
0	0	2	2																																																																																	
1	1	3	3																																																																																	
1	3	2	3																																																																																	
0	1	0	2																																																																																	
0	0	2	2																																																																																	
1	1	3	3																																																																																	
1	1	3	3																																																																																	
0	0	2	2																																																																																	
0	0	2	2																																																																																	
0	0	1	1																																																																																	
0	0	1	1																																																																																	
2	2	3	3																																																																																	
2	2	3	3																																																																																	

特徴

1	2	3		7	4	1		9	8	7
4	5	6	↔	8	5	2	↔	6	5	4
7	8	9		9	6	3		3	2	1

2回の回転で大きく違う状態に遷移できる

スコアをそのまま評価値にするとうまくいかない（ことがある）
要するに、左の状態を目指すより、右の状態を目指す方が手順が短いことがある

今回の問題は、大きく回転させると
Move Penaltyが大きくなるので考慮しなくてよかった

Move Penalty

$$(S-1)^{1.5}$$

$$S=2 \rightarrow MP=1$$

$$S=3 \rightarrow MP=2$$

$$S=4 \rightarrow MP=5$$

$$S=5 \rightarrow MP=8$$

$$S=6 \rightarrow MP=11$$

$$S=7 \rightarrow MP=14$$

任意の 1 つの Cell を移動させる場合

S=2~3 までは MP1 : 距離 1

S=4 以上は、距離以上の MP がかかる

複数まとめて移動させる場合は

S が大きい方が良いので使いみちはある

端からつめる

Cell を移動する際に周りの Cell も移動させて壊すので、端から順に入れていく



112	17	125	20	59	85	62	182	86	193	93	13
119	60	170	143	8	29	40	28	71	1	46	12
154	27	102	191	167	17	33	110	97	148	7	13
159	50	150	83	189	44	15	57	69	163	80	9
131	22	114	55	129	94	31	16	160	84	88	18
75	53	14	5	23	152	30	99	39	2	41	6
187	106	82	49	73	103	11	121	54	87	65	7
32	34	134	186	172	161	141	43	111	26	176	5

小さいケース

DP だと 3×3 くらいまでしかできなさそう
(枝を刈れれば、 4×4 くらいいける…?)

gif をよく眺めてみるとすごく効率が良い
端からつめるが基本方針だけど
それより明らかに効率が良い手順が存在する

焼きなましが困難そうなので、ビームサーチになるが
まともな評価関数を作るのも難しそう (判断難しい)

$N \times N$ どのくらいまでビームサーチでいけるか実験する必要あり
Loc Penalty=0 にする必要はないので
最悪、細かいところは気にしないで良さそう

大きいケース 最大: 40x40

全体をビームサーチは到底できなさそうなので
小さい問題に分解する必要がある

端からつめる必要があるので
最上、最下、最左、最右のどれか
行列単位で順に入れていく…のが普通

807	180	102	369	1187	182	1038	1023	471	1487	204	1022	393	286	1457	513	33	1087	211	338	1461	916	51	515	405	895	420	434	443	268	171	246	1336	1268	1438	842	1320	88	843	73
590	141	1281	994	259	56	178	1122	905	205	149	438	801	1584	1397	965	84	810	1504	934	1340	1215	844	814	587	710	338	49	895	800	450	489	379	577	1589	1170	1217	1484	1134	
65	327	1305	7	647	92	479	1324	803	279	362	210	91	1249	1059	1575	697	1211	1230	440	42	324	1042	115	607	883	1128	277	288	284	287	1124	1581	1392	874	1460	308	431	1026	981
280	380	1415	855	644	840	1129	354	285	835	724	1254	828	1433	1417	1259	82	844	1162	407	1258	1040	1250	283	1384	1541	1491	1030	1449	1197	1172	876	567	1078	833	181	196	875	1426	613
1882	803	885	446	1237	1045	1233	888	1586	1387	874	880	880	709	115	18	869	743	269	808	335	859	82	882	1865	883	1547	838	548	1275	519	701	1189	1585	263	1602	739	787	806	1018
1468	61	1147	214	732	772	820	1430	206	822	153	303	30	1050	70	944	1506	62	709	1252	754	916	24	59	1338	1525	604	887	300	1282	17	828	1141	177	896	307	1100	1432	552	554
1136	877	833	23	189	1120	1379	951	418	897	897	1209	188	488	1483	468	1132	1168	708	85	89	1310	145	1872	1489	1071	38	327	274	302	289	789	146	1210	1880	1811	886	1002	1454	1369
348	1545	142	42	1450	1230	405	1180	187	1586	454	1054	1352	827	1184	456	876	973	488	143	925	458	232	1288	845	854	1125	140	238	732	1826	1308	818	885	275	643	12	1168	847	1490
1689	155	1215	786	778	168	121	812	1276	1383	382	389	848	1331	712	1287	897	479	558	1081	377	247	1091	938	260	1514	338	1040	912	816	789	1302	181	808	985	404	207	426	567	635
535	340	882	1108	850	461	1204	766	507	1445	1444	1080	314	28	1009	851	601	632	280	1436	261	178	1589	26	1488	167	1321	818	547	893	1368	1095	515	1456	257	238	658	445	221	748
714	53	852	1587	736	18	1470	1550	903	438	303	880	1158	1145	1117	322	1084	802	1341	508	1044	482	889	104	6	1202	1270	1482	161	834	777	1174	1512	380	1085	1278	531	1089	682	899
1196	982	1034	947	1127	959	913	443	1287	1089	239	636	1073	1027	197	1003	8	271	1044	200	958	891	1870	19	333	991	75	353	297	136	713	456	1578	1243	1429	1027	80	1245	1333	
27	173	1280	903	424	1272	384	480	728	437	569	327	1350	1444	819	39	164	403	805	1059	1324	918	898	717	1007	1510	1012	438	887	1910	1380	410	122	1232	360	428	126	1309	890	728
386	982	809	1383	1107	72	911	1508	1008	785	484	1007	834	921	791	143	888	1463	584	1223	1463	485	475	326	74	391	1171	1328	818	617	837	1009	638	898	1440	1056	580	1500	1803	60
1041	1395	1359	838	1408	4	810	578	102	810	1310	894	1082	676	929	509	984	20	731	1363	1416	482	567	923	215	1194	975	58	1241	412	1462	400	1315	412	591	29	1089	1484	1163	114
680	1277	212	711	459	264	789	1051	1591	21	89	49	309	1422	1421	384	350	668	771	1079	1552	124	10097	376	1135	719	1096	640	110	1347	1118	1587	1285	338	1049	700	97	1304	972	616
388	1386	1083	504	574	734	664	1284	1272	32	370	338	673	945	315	194	1103	119	1328	1481	13	388	1451	918	1689	1112	1333	324	391	364	174	1445	1371	1402	870	132	270	776	1289	416
457	329	487	1248	1173	780	1332	462	742	1199	1247	337	1429	887	332	802	938	1380	1234	343	1188	1089	828	439	183	414	1573	94	1586	1455	819	525	1488	1219	814	997	1115	546	201	1016
848	849	272	760	1369	1190	583	892	1577	738	1087	716	633	1155	41	282	1382	1162	627	753	715	945	117	300	386	351	835	353	1375	1240	1468	1310	843	682	1389	1047	193	871	286	
490	1137	871	1273	1360	222	1029	395	1056	48	898	718	64	83	1218	947	843	1033	1178	889	1563	1149	801	524	1077	964	370	1185	1052	576	1661	1053	211	998	583	788	150	487	648	584
455	772	1480	1287	1092	304	1380	1005	849	872	1182	748	582	133	1214	811	702	1038	828	641	1280	1111	281	771	1107	425	800	1001	1546	1066	311	1581	778	1322	202	402	1829	127	1186	1287
1376	1084	382	172	120	318	1182	20	822	468	1198	903	1038	1288	805	134	1381	1110	312	831	371	1530	96	1014	430	821	1496	1080	216	432	323	702	254	793	278	723	128	421	1008	
896	1146	720	108	1288	2	112	246	254	478	109	866	385	116	303	899	1489	422	1388	165	1382	1340	1343	1238	688	1138	281	1424	621	1105	268	1584	938	46	838	288	1486	545	274	878
834	1010	1063	1485	37	1630	278	1676	1361	1484	1365	264	403	778	989	1061	1534	1106	1188	138	824	918	1221	748	8435	1684	1327	752	1371	977	804	368	1288	899	623	447	721	470	829	323
1637	1301	282	359	46	600	1088	948	1439	288	1639	34	44	1312	1413	423	667	532	123	296	241	100	1253	1121	665	106	199	13140	962	756	1627	653	1115	927	1181	79	814	1257	190	573
1816	778	1208	998	1128	1274	693	477	846	917	1410	1458	878	846	938	1828	489	1374	964	1040	1810	1020	1387	887	1427	378	851	371	300	1470	1088	834	1314	422	1388	882	453	482	1530	
46	990	1293	440	87	933	469	824	78	801	710	787	228	1242	799	800	1177	941	1517	1813	69	797	1344	886	245	347	830	283	788	262	864	875	1868	1478	137	130	306	788	282	26
1533	1032	872	1179	703	1487	1583	1042	1381	1594	1595	1317	1555	1362	22	1034	305	67	273	868	315	942	1380	563	5	738	1325	878	8	820	258	554	982	501	491	850	1317	746	621	536
1034	1498	188	872	209	537	1533	1634	1000	1349	1236	740	631	1148	1538	828	695	922	628	168	915	715	04	183	812	538	219	1932	483	1231	483	823	1123	612	1176	1176	1569	1437	240	854
95	1023	1430	1031	830	345	232	854	680	1543	699	1517	846	1477	1050	1342	1521	541	580	227	987	346	07	1036	320	1520	684	790	199	783	915	978	1378	680	1204	1447	532	310	480	1027
691	680	886	1114	226	611	910	289	1226	751	1327	190	366	76	220	1222	305	1388	747	105	683	1299	294	586	191	1004	320	1479	796	1384	489	876	876	387	1339	1532	472	1070	158	1288
1340	340	841	1062	427	111	14	1646	125	1315	1320	1412	267	1431	811	588	877	102	433	781	444	1448	1075	1244	287	373	89	25	200	868	584	1386	1468	1140	483	16	881	1271	1212	643
411	1377	1094	1348	618	1021	1642	796	852	138	387	208	486	481	11	1465	1150	725	887	518	842	889	280	383	1263	1072	428	947	1867	1383	617	517	1582	1018	889	86	1370	81	1507	
1449	787	192	1156	1334	47	849	1426	528	817	1542	1164	434	571	1192	1441	793	1330	16	629	923	328	806	882	1198	429	764	899	280	241	239	556	1594	1303	80	131	1013	1411	1407	
224	253	160	1201	1468	820	451	1473	885	581	1545	484	787	1262	347	874	879	255	1220	694	381	741	888	9	761	1548	784	118	501	1387	688	1285	1335	77	735	1019	930	383	1448	753
1689	1237	186	681	837	3	248	196	231	473	498	722	1430	342	961	474	1022	1565	264	794	1408	135	1430	896	162	548	953	417	2339	54	1024	963	1594	1044	321	925	848	1205	93	
478	1108	249	1318	638	1068	31	1204	1043	1108	1587	1463	553	181	129	1078	964	840	1338	908	1																			








優勝者考察

Psyho さんが 2 位以下を大きく離して優勝

1 位 : 99

2 位以下 : 91 以下

score から約 1 割くらい penalty が低い

Rank			User		Score		
Final 	Provisional 	Rating 	Username 	Final 	Provisional 	Time 	
1	1	2749	Psyho	99.02031	98.94882	30 Apr 2020 00:42:19	History (3) >
2	2	2590	tomerun	91.37678	89.67719	29 Apr 2020 20:37:29	History (2) >
3	3	2666	Milanin	88.87623	88.80232	29 Apr 2020 10:49:00	History (3) >
4	5	2264	Daiver19	88.41422	86.19065	30 Apr 2020 02:59:50	History (22) >
5	4	1889	simanman	88.05218	86.9307	30 Apr 2020 01:25:05	History (42) >
6	6	2185	yowa	84.77582	85.6745	29 Apr 2020 10:57:36	History (4) >
7	7	2398	imazato	84.51998	82.93959	29 Apr 2020 18:55:01	History (18) >
8	9	2195	EvbcFfp1XB	84.48671	82.13889	29 Apr 2020 17:55:42	History (6) >
9	8	2265	sullyper	82.60673	82.58648	30 Apr 2020 00:42:38	History (11) >
10	10	1763	sas4eka	81.96	80.79994	29 Apr 2020 22:44:35	History (11) >

Psyho さん 優勝

Forums: <https://apps.topcoder.com/forums/?module=Thread&threadID=955063&start=0&mc=3#2406205>

source code: <https://pastebin.com/SqvXmJ8S>

That was a really nice problem with some interesting complexity. Probably one of the very few ones where I had to use pen & paper.

Key observations:

P doesn't matter, you should (almost) always fully solve the board. The baseline solution is to just move numbers one by one, row by row. There are probably some cases for small boards when fully solving is suboptimal but I decided to ignore those.

The penalty function is a little bit unusual in that 2x2 & 3x3 are really cheap, 4x4-6x6 are expensive and then it gets cheaper the larger they get.

Most of the moves are independent of each other and so the number of states explode quickly. Vanilla beam search will struggle here with anything bigger than tiny.

I use a divide-and-conquer (puzzle part) solution with beam search (pathfinding part) as a building block.

I have two basic cost functions and beam search is run with one of them. Standard manhattan distance (with potential non-linear scaling) and "split" function where every number is converted to a binary value that tells which subrectangle it belongs to. The value is the distance to the correct subrectangle. The second one greatly simplifies the complexity during splitting and makes it easier to detect duplicated states. There's a variation of latter one with simultaneous 4-way split.

Building blocks:

solve_full>: Solve NxM. Small -> beam search. Medium -> do a special 4-way split + 4 x standard beam search. Big -> run solve_split.

solve_split>: Split NxM. Small -> beam search (with "split" cost function). Big Square-ish -> run enable_split and solve 2 subrectangles separately. Rectangle -> Assume NxM (where N>M). Divide roughly into NxM/2 and NxM/2, solve separately and join them by a big180TM MxM rotation in the middle.

enable_split: Make sure that two subrectangles with solved separately. In order to enable that, they need to have an exact number of each type of cell. This is achieved by a counting surplus/demand for each subrectangle and then by applying our standard binary labels to missing/surplus cells and running our standard beam search with split function.

It may sound a bit confusing but it's actually quite clear when you look in the code.

The key parts were the parallel split (that is possible thanks to enable_split) to reduce the overall size of grid search runs. And the big180TM move to join results from two different subproblems - that way we never really have to cover huge distances for any numbers outside of the big moves.

I'd imagine there are many different overall strategies here that all yield somewhat similar results.

Random interesting bits:

Since I haven't included any "combo moves" in my state transition, I rely on decent beam width in order to arrive at the solution. Because of this and because of unpredictability of my execution time, I believe I use only half allotted time or less. Doubling beam width gives me ~1.5%.

It's quite probable that I will fail few cases since I have no time checks and no fallback if one of the beam search runs fails to find solution.

Somewhat silly/cool idea if someone had way too much time: after developing a set of approaches for every NxM rectangle and gathering stats about them (average score & running times), you can do DP over all of that in order to find a nearly-optimal way of solving each problem instance.

Using dist^{1.75} instead of dist for normal beam search gave me ~1% improvement.

I only do 2x2 & 3x3 rotations outside of big180TMs. With 4x4 I actually get worse score. This may hint that there's a room for improvement for finding a better penalty function

DeepL 翻訳

興味深い複雑さを持った本当に素晴らしい問題でした。おそらく、ペンと紙を使わなければならなかった数少ない問題の一つです。

重要な観察事項です。

P は重要ではありません。基本的な解法は、数字を一行ずつ動かしていくことです。小さな盤の場合、完全に解くことが最適ではない場合もあるでしょうが、私はそれを無視することにしました。

ペナルティ関数は、 2×2 と 3×3 が本当に安く、 4×4 - 6×6 が高く、大きくなるほど安くなるというちょっと変わったものです。

ほとんどの手はお互いに独立しているので、状態の数はすぐに爆発します。バニラのビームサーチは、小さいものよりも大きいものには苦戦するでしょう。

私は、ビームサーチ（経路探索の部分）を構成要素として、分割と征服（パズルの部分）のソリューションを使用しています。

私は 2 つの基本的なコスト関数を持っていて、ビームサーチはそのうちの 1 つで実行されます。標準的なマンハッタン距離（潜在的非線形スケーリングを含む）と "分割" 関数では、すべての数値がどのサブレクタングルに属するかを示す二進法の値に変換されます。値は、正しい長方形までの距離です。2 番目のものは、分割時の複雑さを大幅に単純化し、重複した状態の検出を容易にします。後者のバリエーションとして、同時 4 分割があります。

ブロックを構成しています。

`solve_full`: NxM を解く。Small -> ビームサーチ。Medium -> 特別な 4 分割 + $4 \times$ 標準ビームサーチを行う。大 -> `solve_split` を実行。

`solve_split`: NxM を分割。小 -> ビームサーチ ("split" のコスト関数を使って)。Big Squar-ish -> `enable_split` を実行し、2 つのサブレクタングルを別々に解く。長方形 \rightarrow NxM を仮定（ここでは $N > M$ ）。 $N \times M / 2$ と $N \times M / 2$ に大まかに分けて、別々に解いて、真ん中で `big180TM` の $M \times M$ 回転で結合する。

`enable_split` を有効にします。2 つのサブレクタングルが別々に解けていることを確認してください。それを有効にするためには、彼らはセルの各タイプの正確な数を持っている必要があります。これは、各サブレクタングルの余剰 / 需要をカウントし、不足 / 余剰セルに標準的なバイナリラベルを適用して、スプリット関数を使って標準的なビームサーチを実行することによって達成されます。

少し混乱しているように聞こえるかもしれませんが、実際にはコードを見ると非常にはっきりしています。

重要な部分は、グリッド検索の全体的なサイズを小さくするための並列分割（これは `enable_split` のおかげで可能です）です。そして、2 つの異なるサブプロブレムからの結果を結合するための `big180TM` の移動です。

全体的な戦略はいろいろあると思いますが、どれも似たような結果になります。

ランダムに面白いことを紹介します。

私は状態遷移に「コンボの動き」を含めていないので、解決策にたどり着くためには、適切なビーム幅に頼っています。これと実行時間の予測不可能性のため、半分以下の時間しか使っていないと思います。ビーム幅を 2 倍にすると 1.5% 程度になります。

ビーム幅を 2 倍にすると 1.5% 程度になりますが、タイムチェックがないので、失敗することはほとんどありません。

もし時間がありすぎる人がいたら、ちょっとばかばかしい / クールなアイデアがあるかもしれません：すべての $N \times M$ 矩形に対してアプローチのセットを開発し、それらについての統計情報（平均スコアと実行時間）を集めた後で、各問題のインスタンスを解くためのほぼ最適な方法を見つけるために、それらすべての DP を行うことができます。

通常のビーム探索では `dist` の代わりに `dist^1.75` を使うことで 1% 程度の改善が得られました。

私は `big180TMs` 以外では 2×2 と 3×3 の回転しかしていません。 4×4 では、実際にはスコアが悪くなります。これは、より良いペナルティ関数を見つけるための改善の余地があることを示唆しているかもしれません。

小さく分割する

112	117	125	20	59	85	182	161	86	193	93	137	92	68
119	60	170	143	8	28	62	110	71	1	46	124	35	61
154	27	102	191	167	40	33	69	97	148	7	136	48	56
159	50	150	83	189	29	17	44	121	163	80	9	118	153
131	22	114	55	129	94	165	57	99	39	160	181	169	127
75	53	14	152	103	172	138	16	141	31	84	63	196	90
187	106	82	23	73	11	104	144	15	30	88	77	158	38
32	34	134	5	49	186	192	51	111	2	135	188	128	72
108	177	180	147	42	113	65	164	120	176	179	190	155	21
19	81	37	133	156	151	41	43	105	54	87	174	140	149
146	70	12	175	185	173	26	78	184	101	79	4	10	66
168	166	107	91	116	67	123	25	18	3	24	96	98	95
178	89	171	6	100	47	76	45	183	195	194	142	74	13
132	58	139	126	52	36	64	122	145	162	130	157	109	115

29	49	1	2	5	86	85	50	28	80	36	97	92	68
20	33	17	16	75	7	35	55	82	64	83	93	65	9
15	45	59	31	47	91	71	8	81	37	69	11	56	41
60	4	34	18	43	90	46	40	27	52	70	14	84	38
57	44	77	87	61	48	30	24	53	12	54	78	23	79
58	72	62	76	74	63	88	25	94	66	39	51	42	13
32	19	73	89	6	21	3	67	22	95	26	10	98	96
187	117	114	159	127	119	158	168	108	121	152	134	153	137
173	170	143	131	102	155	118	191	112	182	167	124	196	193
144	129	189	161	184	174	128	106	150	107	111	181	163	176
147	105	185	172	160	169	145	154	177	120	110	136	148	140
133	99	186	103	104	101	130	139	166	125	122	180	138	149
175	146	141	116	157	100	188	178	162	192	123	126	151	135
132	171	156	113	183	142	115	164	190	165	109	179	195	194

14x14 → 7x7 * 4 に分割

後付考察

端からつめる場合、多くの部分が壊れる
→ 効率が悪くなりがち

ビームサーチ解は壊れる部分を緩和している
小さい問題に分割する方法は
ビームサーチが適用可能な箇所が増えるので
その分、 **score** が改善している…気がする

小さく分割するコストがどれくらいになるのか
少し想像しづらい。実験するしかないか？

1	2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27	28
29	30	31	32	33	34	35	36	37	38	39	40	41	42
43	44	45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96	97	98
99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126
127	128	129	130	131	132	133	134	135	136	137	138	139	140
141	142	143	144	145	146	147	148	149	150	151	152	153	154
155	156	157	158	159	160	161	162	163	164	165	166	167	168
169	170	171	172	173	174	175	176	177	178	179	180	181	182
183	184	185	186	187	188	189	190	191	192	193	194	195	196

詳細：コード読み中…

7x7 以下は分割せずそのまま解く
評価値は、 $\text{dist}^{1.75}$

「まわしてそろえる」が部分問題として出てくる

時間を使い切っていない
ビームサーチの時間管理が面倒
何らかの答えをもっておきたいが、ビームサーチが not 頻出なので枯れない

答えを知った後だと、極めて自然なアプローチに感じる

Link

<https://togetter.com/li/1501039>

<https://bitbucket.org/tomerun/marathon/src/master/MM117/memo.md>