Nama : Timothy Hosia Budianto

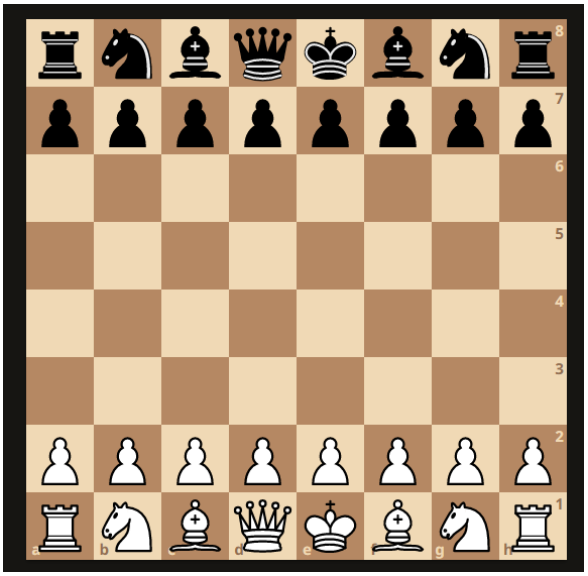
NRP : 5025211098

Kelas : Perancangan dan Analisis Algoritma

**SPOJ BATTLECRY | Battle of The Bastards | King vs Rooks**

**Analisis Permasalahan :**

Soal berjudul SPOJ BATTLECRY -Battle of The Bastards – King vs Rooks ini merupakan sebuah soal kompleks bertiple multiple test cases. Soal ini merupakan adapatasi dan modifikasi dari permainan catur pada umunya, dimana tujuan permainan adalah untuk membuat raja lawan ” terancam termakan” dan tidak dapat bergerak kemana mana lagi untuk menghindari “termakan”. Pemain pada umumnya pada permainan catur biasa akan bermain berganti-gantian, tiap piece di catur memiliki peraturan khusus dimana piece tersebut dapat bergerak untuk memakan piece lawan. Pada catur tiap piece juga diberikan warna berbeda yaitu hitam dan putih untuk menandakan tim. Papan permainan catur pada umumnya akan seperti berikut :



Sumber Board editor • lichess.org

Nah pada soal ini sendiri aka nada modifikasi – modifikasi yang menghasilkan peraturan sebagai berikut :

1. Permainan dilakukan pada papan berukuran $N \times N$

2. Permainan akan dimainkan dengan total 4 bidak catur, yaitu bidak raja putih, bidak raja hitam, serta 2 bidak benteng putih. Keempat bidak tersebut harus ditempatkan di posisi yang berbeda.

3. Raja dapat bergerak satu kotak ke arah manapun, baik horizontal, vertikal, atau diagonal dengan catatan bahwa raja tidak boleh keluar dari area atau papan permainan, tidak berada dalam posisi yang sama dengan bidak lain, serta tidak membuat raja dalam posisi bahaya (kondisi check).

4. Bidak benteng dapat bergerak ke semua kotak di rank atau file yang sama dengan catatan tidak boleh melewati bidak lain dan tidak boleh bergerak ke posisi yang telah ditempati oleh bidak sewarna lain.

5. Sebuah bidak $X$ diancam oleh bidak lawan $Y$ apabila bidak $Y$ dapat bergerak ke posisi bidak $X$ dalam 1 langkah legal.

6. Bidak $X$ dapat menangkap bidak lawan $Y$ apabila bidak $X$ dapat bergerak ke posisi bidak $Y$ dalam 1 langkah legal. Bidak $Y$ akan dikeluarkan dari permainan dan posisinya ditempati oleh bidak $X$. Jika bidak $X$ adalah raja maka bidak $X$ tidak bisa menangkap bidak $Y$ jika bidak tersebut dijaga oleh bidak lain.

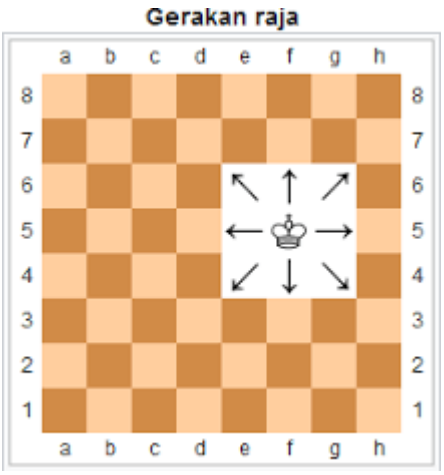
7. Kedua bidak raja (putih dan hitam) tidak bisa berada di kotak-kotak bersebelahan, baik horizontal, vertikal, ataupun diagonal.

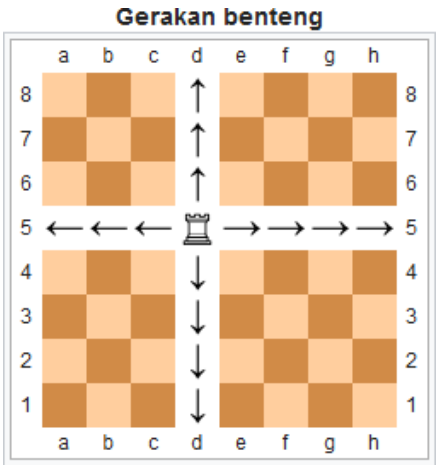
8. Tujuan dari permainan adalah menempatkan keempat bidak catur pada posisi tertentu sehingga raja hitam diancam dan tidak bisa menghindari ancaman tersebut pada saat giliran pemain hitam, atau kata lain, raja hitam dalam kondisi checkmate.

Susunan papan tidak harus legal menurut aturan catur. Ini berarti bahwa konfigurasi papan yang tidak layak dalam semua keadaan dalam permainan catur dianggap sebagai konfigurasi yang valid untuk permainan itu.

Pada soal ini bisa diberikan T dengan range dari 1 – 20. Untuk setiap testcase akan menghasilkan sebuah papan N dengan range 1 – 10 pangkat 5. Setiap inputan integer N akan menghasilkan sebuah papan dengan size n x n. Contohnya jika inputan N adalah 8 akan

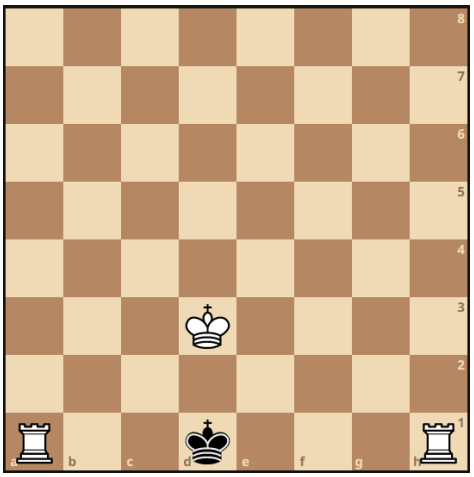
menhasilkan sebuah papan yang berukuran 8 x 8 dan seterusnya. Selanjutnya kita akan diminta mengoutputkan berapa banyak kemungkinan penempatan 1 raja hitam 1 raja putih 2 benteng putih. Semua chess pieces tersebut akan ditempatkan sebgaimana rupa hingga menyebabkan kondisi raja hitam checkmate dan tidak ada yang bisa dia lakukan untuk keluar dari posisi terancam baik dengan berpindah ke kotak aman maupun menggunakan chess pieces untuk memakan chess pieces lainya. Bidak raja dapat bergerak ke segala arah sebanyak 1 kotak tiap 1 kali turn, sedangkan bidak benteng dapat bergerak secara horizontal dan vertical dengan jumlah kotak tak terbatas.

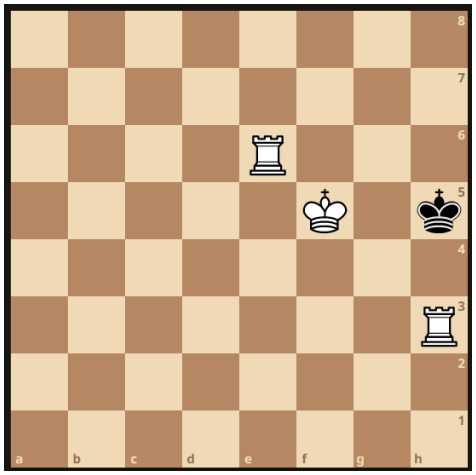



Sumber : [Nama, Langkah, dan Gerakan Biji / Buah Catur - ATURAN PERMAINAN]

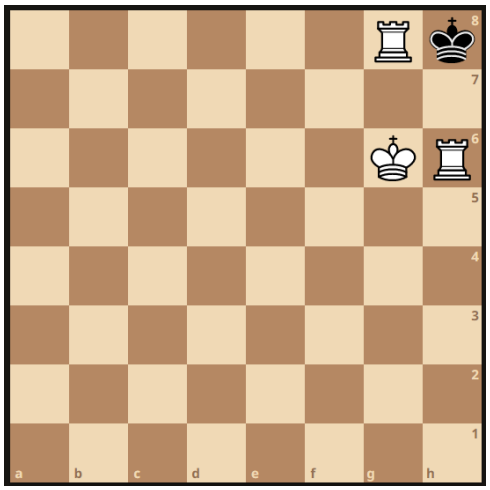
Berikut adalah beberapa contoh konfigurasi posisi yang valid di N 8 x 8 dari 72392 konfigurasi yang bisa



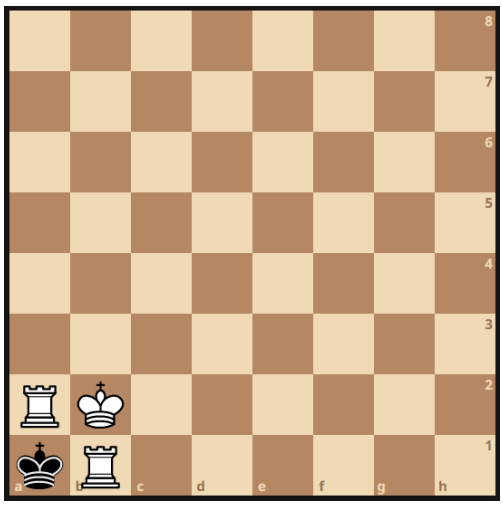


Sumber [Board editor • lichess.org]

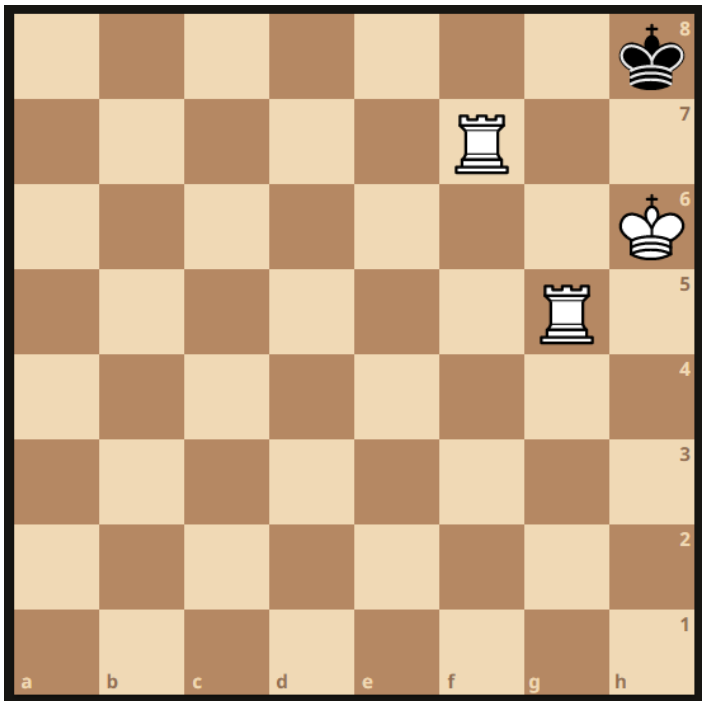
Berikut adalah beberapa  contoh konfigurasi posisi yang  tidak valid di N 8 x 8



Tidak valid karena raja hitam bisa memakan benteng putih di g1



Tidak valid karena raja tidak boleh saling bersebelahan



Tidak valid karena raja hitam tidak terancam

Selain itu pada n size 1 dan 2 akan mengoutputkan 0 karena tidak menghasilkan hasil yang valid. Soal ini memiliki limit sebagai berikut :

| | |
|---|---|
| Time limit: | 5s |
| Source limit: | 50000B |
| Memory limit: | 1536MB |

**Abstraksi :**

Apabila soal diselesaikan dengan cara brute force dengan kompleksitas O (TN^8) akan mendapatkan verdict TLE ( time limit exceeded ) karena program akan berjalan selama kurang lebih 10^33 detik.

Pertama kita akan mencoba menggunakan metode brute force. Implementasi brute force adalah mengecheck semua kemungkinan konfigurasi yang akan menghasilkan bilangan berpola. Setelah itu kita akan melakukan implementasi penyusunan formula yaitu Menyusun formula sesuai dengan barisan bilangan berpola yang telah dihasilkan.

Implementasi penyusunan formula akan dibagi menjadi 2. Implementasi pertama menggunakan implementasi penyusunan **linier rekuren homogen** menggunakan **algoritma berlekamp-massey.** Yang kedua implementasi penyusunan **formula polynomial** menggunakan **interpolasi lagrange**. Selain itu kita akan menggunakan konsep dynamic programming untuk mempersingkat waktu dan penyimpanan.

Metode bruteforce dilakukan dengan mencoba semua kemungkinan konfigurasi penempatan keempat bidak pada papan catur berukuran N x N. Total konfigurasi yang dapat dibentuk dapat dipandang sebagai permasalahan kombinatorik, yaitu banyaknya permutasi pemilihan posis dari N x N petak untuk empat bidak catur tanpa perulangan. Hasil permutasi akan dibagi 2 karena kedua benteng putih adalah identic sehingga pertukaran posisi pada kedua benteng dianggap sebagai sebuah konfigurasi yang sama. Solusi perhitungan banyaknya total konfigurasi berdasarkan ukuran papan catur N akan dinotasikan dengan :

$$f(N) = \frac{P(N \times N, 4)}{2}$$

Dengan implementasi brute force diatas akan menghasilkan hasil sebagai berikut :

| Ukuran Papan Catur (N) | Total Konfigurasi Valid | Ukuran Papan Catur (N) | Total Konfigurasi Valid |
|---|---|---|---|
| 1 | 0 | 11 | 414936 |
| 2 | 0 | 12 | 664504 |
| 3 | 232 | 13 | 1022452 |
| 4 | 1432 | 14 | 1520872 |
| 5 | 5188 | 15 | 2197648 |
| 6 | 14536 | 16 | 3096936 |
| 7 | 34464 | 17 | 4269644 |
| 8 | 72392 | 18 | 5773912 |
| 9 | 138652 | 19 | 335559 |
| 10 | 246968 | 20 | 2708695 |

Sumber semua ilustrasi penjelasan dari ppt mas Aldo Yaputra Hartono

Setelah kita mengetahui beberapa barisan berpola kita dapat melakukan implementasi penyusunan formula.

Pertama kita akan menggunakan implementasi penyusunan menggunakan algoritma **berlekamp-massey.** Pada strategi ini dilakukan *precompute* terhadap barisan bilangan berpola yang telah dihasilkan oleh metode bruteforce guna mendapatkan semua koefisien pada linear homogen dengan variasi banyaknya bilangan berpola yang berbeda.

Pada algoritma ini menerapkan paradigma Dynamic Programming dengan pendekatan *bottom-up* untuk menyimpan dan memperbarui nilai koefisien linear rekuren homogen pada tiap perulangan. Berikut adalah pseudocode metode **berlekamp-massey**

Pseudocode 3. 9 Fungsi *berlekampMassey* Algoritma *Berlekamp-Massey*

```
    Input : s
    Output: C
1   n ← s.size()
2   l ← 0
3   m ← 0
4   B(n)
5   C(n)
6   ld ← 1
7   B[0] ← 1
8   C[0] ← 1
9   for i ← 0 to n do
10    d ← s[i] % MOD
11    for j ← 1 to l+1 do
12      d ← (d + C[j] * s[i-j]) % MOD
13    end
14    if (d == 0) then
15      m ← m + 1
16    else
17      T ← C
18      coef ← d * bigmod(ld, MOD-2) % MOD
19      for j ← m to n do
20        C[j] ← (C[j] - coef * B[j-m]) % MOD
21      end
22      if (2 * l ≤ i) then
23        l ← i - l + 1
24        B ← T
25        m ← 1
26        ld ← d
27      else
28        m ← m + 1
29      end
30    end
31  end
32  C.resize(l+1)
33  C.erase(C.begin())
34  for i ← 0 to l do
35    C[i] ← (MOD - C[i]) % MOD
36    if (2 * C[i] > MOD) then
37      C[i] ← C[i] - MOD
38    end
39  end
40  return C
```

Berikut adalah penjelasan alur metode berlekamp-massey menggunakan sebuah testcase

| s | 0 | 0 | 232 | 1432 | 5188 |
|---|---|---|---|---|---|
| B | 1 | 0 | 0 | 0 | 0 |
| C | 1 | 0 | 0 | 0 | 0 |

Disini S melambangkan sequence, kita akan mengambil contoh illustrasi dengan mengambil 5 bilangan berpola awal. Dimana niai b akan menyimpan koefesien terbaik sebelumnya. Dan nilai C adalah nilai sebenarnya yang akan kita pakai.

$$n = 5$$
$$l = 0$$
$$m = 0$$
$$ld = 1$$

Nilai **n** adalah 5 yang menandakan 5 buah bilangan berpola, **l** adalah ukuran dari koefesien yang diinisialisasi nilai 0, dan **m** adalah jumlah stepnya, sedangkan **ld** adalah last delta yaitu selisih nilai sekarang dan nilai sebelumya. Selain itu kita juga menyimpan nilai **d** yang berguna untuk menyimpan nilai hasil. Berikut adalah nilai nilai tiap variable dengan i menandakan iterasi keberapa

$$n = 5$$
$$l = 0$$
$$m = 0$$
$$ld = 1$$

$$i = 0$$

$$d = s[i] = s[0] = 0$$

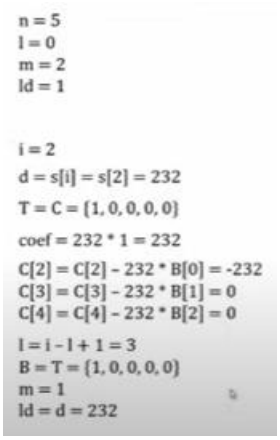
$$m = m + 1 = 0 + 1 = 1$$

$$i = 1$$

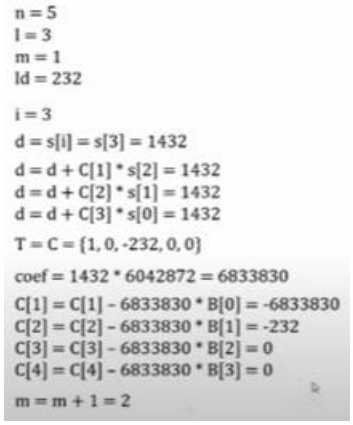
$$d = s[i] = s[1] = 0$$

$$m = m + 1 = 1 + 1 = 2$$

Kemudian kita akan memperbarui nilai m nya dan masuk ke iterasi selanjutnya




Didapatkan nilai d sementaranya yaitu 232, dan karena nilai l masih 0 kita akan lanjut. Karena nilai d nya tidak 0, diliat dari lline 17 kita menyimpan kita akan menyimpan koefisien sekarang pada nilai t yaitu temporary. Lalu kita akan menghitung nilai koefisiennya yaitu rasio perkaliannya. Jadi kita dalam algoritma ini kita akan melakukan penyesuaian dan pembaruan nilai koefesien sesuai dengan nilai rasio yang didapat dari **nilai delta diibagi last delta**. Tapi karena disini menggunakan operasi modulo juga, maka perhitungan pembagian modulo bisa digunakan operasi **invers modulo**. Jadi bisa delta dikali invers modulo dari last delta akan didapatkan nilai coefnya 232. Nilai coef ini akan menjadi rasio pengkali untuk memperbarui nilai coefisiennya. Setelah dilakukan update nilai koefesien akan dilakukan update untuk variable lainnya. Dengan metode yang sama akan dilakukan dengan iterasi selanjutnya




Di tiap iterasi akan dilakukan pengecheckan oleh line 22 apakah jumlah dari coefesien dikali 2 apakah akan kurang dari sama dengan step saat ini, jika iya kita akan melakukan update dari beberapa nilai variable, jika tidak kita hanya perlu menicrement nilai step (m). Lanjut pada iterasi berikutnya

```
n = 5
l = 3
m = 2
ld = 232

i = 4
d = s[i] = s[4] = 5188
d = d + C[1] * s[3] = -1775383
d = d + C[2] * s[2] = -1829207
d = d + C[3] * s[1] = -1829207
T = C = {1, -6833830, -232, 0, 0}
coef = (-1829207) * 6042872 = -7126451
C[2] = C[2] - (-7126451) * B[0] = 7126219
C[3] = C[3] - (-7126451) * B[1] = 0
C[4] = C[4] - (-7126451) * B[2] = 0
m = m + 1 = 3
```

Setelah iterasi ke 5 kita sudah mendapatkan nilai yang kita minta. Maka pada akhirnya kita akan memiliki nilai koefesien seperti berikut

| | | | | | |
|---|---|---|---|---|---|
| s | 0 | 0 | 232 | 1432 | 5188 |
| B | 1 | 0 | 0 | 0 | 0 |
| C | 1 | -6833830 | 7126219 | 0 | 0 |

Untuk koefesien diatas masih bernilai “mentah” dan perlu dilakukan sedikit pengolahan disini untuk mendapatkan nilai koefesien yang sesungguhnya yang akan menhasilkan hasil sebagai berikut

| Banyak Bilangan Berpola | Koefisien Linear Rekuren Homogen | Akurasi (%) |
|---|---|---|
| 1 | - | 10 |
| 2 | - | 10 |
| 3 | 0, 232, 0 | 13 |
| 4 | -506203, 232, 0 | 20 |
| 5 | -506203, 213814, 0 | 25 |
| ... | ... | ... |
| 16 | 6, -15, 20, -15, 6, -1, 0, 0 | 100 |
| 17 | 6, -15, 20, -15, 6, -1, 0, 0 | 100 |
| 18 | 6, -15, 20, -15, 6, -1, 0, 0 | 100 |
| 19 | 6, -15, 20, -15, 6, -1, 0, 0 | 100 |
| 20 | 6, -15, 20, -15, 6, -1, 0, 0 | 100 |

Bisa dilihat hasil sesungguhnya tiap koefesien seperti table diatas. Untuk table akurasi digunakan dengan perbandingan nilai sesuai yang dihasilkan dari metode bruteforce. Bisa dilihat bahwa minimal diperlukan iterasi hingga 16 bilangan berpola untuk mendappatkan hasil 100 persen yang meenghasilkan 8 buah koefesien. Berikut adalah formula liner rekuren homogen yang dihasilkan, dengan a0 hingga a7 sesuai dengan sample outputnya:

Formula linear rekuren homogen :

$$a_0 = 0$$
$$a_1 = 0$$
$$a_2 = 232$$
$$a_3 = 1432$$
$$a_4 = 5188$$
$$a_5 = 14635$$
$$a_6 = 34464$$
$$a_7 = 72392$$
$$a_n = 6a_{n-1} - 15a_{n-2} + 20a_{n-3} - 15a_{n-4} + 6a_{n-5} - a_{n-6}$$

Selanjutnya kita akan menggunakan metode interpolasi lagrange. Untuk strategi ini dimulai dengan pengisian formula polynomial dimana penyusunan ini akan melibatkan 6 buah bilangan berpola yang dimulai dari papan catur berukuran 3 hingga 8 dan disusun Kembali sehingga membentuk pasangan x dan f(x) sebagai berikut

| Ukuran Papan Catur | x | f(x) |
|---|---|---|
| 3 | 0 | 232 |
| 4 | 1 | 1432 |
| 5 | 2 | 5188 |
| 6 | 3 | 14536 |
| 7 | 4 | 34464 |
| 8 | 5 | 72392 |

Pada interpolasi ini akan menerapkan paradigma *dynamic programming* dengan pendekatan *bottom-up* untuk menyimpan dan memperbarui nilai koefesien formula polynomial pada tiap perulangan. Pada metode ini akan menghasilkan formula umum sebagai berikut

$$f_n(x) = \sum_{i=0}^{n-1} L_i(x) f(x_i)$$
$$L_i(x) = \prod_{j=0, j \neq i}^{n-1} \frac{x - x_j}{x_i - x_j}$$

Formula diatas merupakan gabungan dari beberapa fungsi. Ada fungsi L yaitu suatu operasi perkalian polynomial dari pasangan x dan f(x). F(x) merupakan banyaknya konfigurasi yang valid. Selanjutnya alur dan pesudocode dari metode ini adalah sebagai berikut

Pseudocode 3. 14 Fungsi *lagrange* Interpolasi *Lagrange*

```
    Input : xPoints, yPoints, nSize
    Output: coef
1   coef(nSize, 0)
2   for i ← 0 to nSize do
3     new_coef(nSize, 0)
4     if (i == 0) then
5       new_coef[0] ← -xPoints[1] / (xPoints[i] - xPoints[1])
6       new_coef[1] ← 1.0 / (xPoints[i] - xPoints[1])
7       startIndex ← 2
8     else
9       new_coef[0] ← -xPoints[0] / (xPoints[i] - xPoints[0])
10      new_coef[1] ← 1.0 / (xPoints[i] - xPoints[0])
11      startIndex ← 1
12    end
13    for j ← startIndex to nSize do
14      if (i ≠ j) then
15        for k ← nSize-1 to 0 do
16          new_coef[k] ← (new_coef[k] * (-xPoints[j]) + new_coef[k-1]) /
            (xPoints[i] - xPoints[j])
17        end
18        new_coef[0] ← (new_coef[0] * (-xPoints[j])) / (xPoints[i] - xPoints[j])
19      end
20    end
21    for j ← 0 to nSize do
22      coef[j] ← coef[j] + new_coef[j] * yPoints[i]
23    end
24  end
25  return coef
```

Dari fungsi lagrange ini bisa dibagi menjadi beberapa bagian seperti berikut

Dapatkan nilai sementara koefisien pertama dan kedua

Lakukan perkalian polinomial dan perbarui nilai koefisien sementara

Lakukan penjumlahan koefisien untuk mendapatkan koefisien akhir

```
4   if (i == 0) then
5     new_coef[0] ← -xPoints[1] / (xPoints[i] - xPoints[1])
6     new_coef[1] ← 1.0 / (xPoints[i] - xPoints[1])
7     startIndex ← 2
8   else
9     new_coef[0] ← -xPoints[0] / (xPoints[i] - xPoints[0])
10    new_coef[1] ← 1.0 / (xPoints[i] - xPoints[0])
11    startIndex ← 1
12  end
13  for j ← startIndex to nSize do
14    if (i ≠ j) then
15      for k ← nSize-1 to 0 do
16        new_coef[k] ← (new_coef[k] * (-xPoints[j]) + new_coef[k-1]) /
          (xPoints[i] - xPoints[j])
17      end
18      new_coef[0] ← (new_coef[0] * (-xPoints[j])) / (xPoints[i] - xPoints[j])
19    end
20  end
21  for j ← 0 to nSize do
22    coef[j] ← coef[j] + new_coef[j] * yPoints[i]
23  end
24  end
25  return coef
```

Berikut hasil dari perkalian polynomial dari L0 sampai L5

Hasil perkalian polinomial :

$$L_0(x) = \prod_{j=0, j\neq 0}^{5} \frac{x - x_j}{x_0 - x_j} = \frac{x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120}{-120}$$

$$L_1(x) = \prod_{j=0, j\neq 1}^{5} \frac{x - x_j}{x_1 - x_j} = \frac{x^5 - 14x^4 + 71x^3 - 154x^2 + 120x}{24}$$

$$L_2(x) = \prod_{j=0, j\neq 2}^{5} \frac{x - x_j}{x_2 - x_j} = \frac{x^5 - 13x^4 + 59x^3 - 107x^2 + 60x}{-12}$$

$$L_3(x) = \prod_{j=0, j\neq 3}^{5} \frac{x - x_j}{x_3 - x_j} = \frac{x^5 - 12x^4 + 49x^3 - 78x^2 + 40x}{12}$$

$$L_4(x) = \prod_{j=0, j\neq 4}^{5} \frac{x - x_j}{x_4 - x_j} = \frac{x^5 - 11x^4 + 41x^3 - 61x^2 + 30x}{-24}$$

$$L_5(x) = \prod_{j=0, j\neq 5}^{5} \frac{x - x_j}{x_5 - x_j} = \frac{x^5 - 10x^4 + 35x^3 - 50x^2 + 24x}{120}$$

Kemudian untuk formula polynomial yang dihasilkan adalah sebagai berikut

Formula polinomial :

$$f_6(x) = \sum_{i=0}^{5} L_i(x) f(x_i)$$

$$f_6(x) = 4x^5 + \frac{124}{3}x^4 + 158x^3 + \frac{1364}{3}x^2 + 542x + 232$$

$$f_6(x) = 4x^5 + 2446719x^4 + 158x^3 + 4893810x^2 + 542x + 232$$

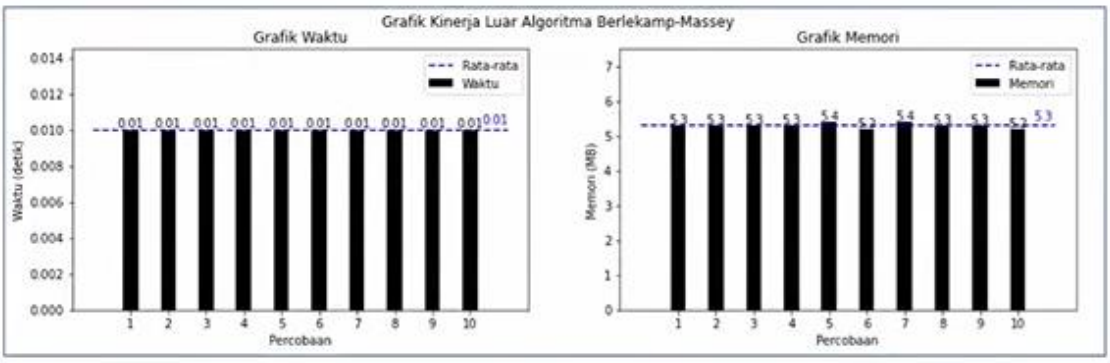
## Uji Coba dan Analisis

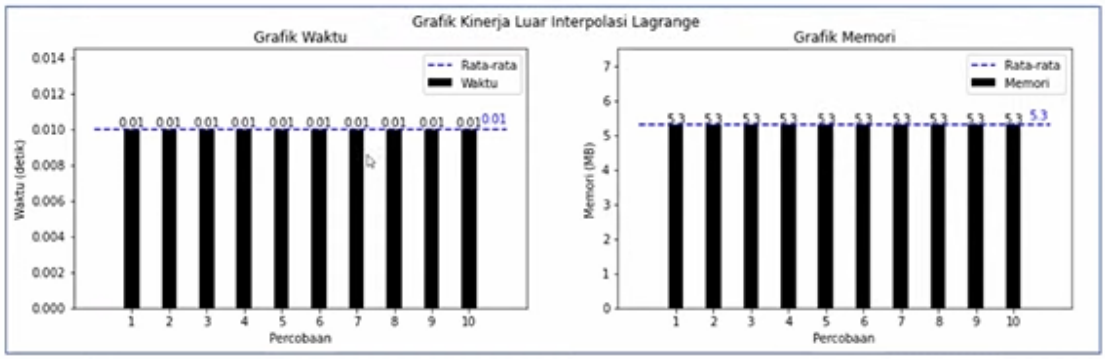
Dari kedua formula itu akan dilakukan uji coba dan analisis dengan metode pengujian sebagai berikut yang dilakukan oleh mas Aldo Yaputra

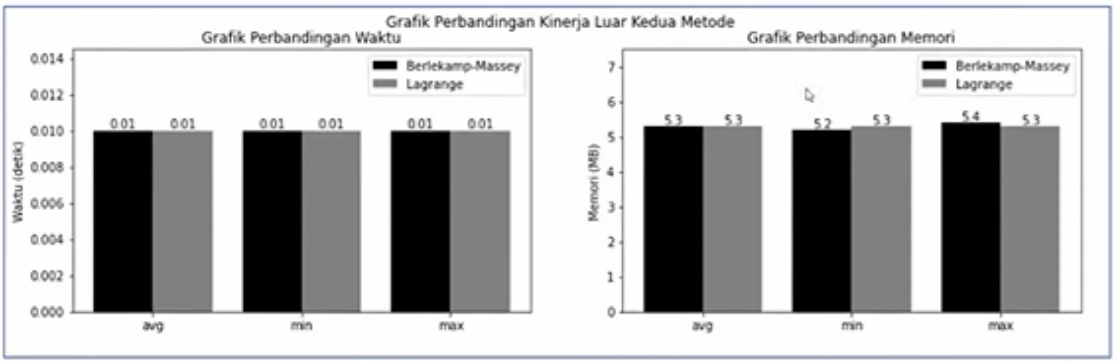
Metode pengujian yang dilakukan :

- Pengujian luar
  - Menggunakan *online judge* untuk menguji kebenaran dan kinerja program.
  - Dilakukan dengan mengirimkan kode program ke situs penilaian daring SPOJ sebanyak 10 kali untuk masing-masing metode penyusunan formula.
- Pengujian lokal
  - Menggunakan mesin yang digunakan dalam pengembangan untuk mengukur kinerja program.
  - Total kasus uji = 100000 kasus (keseluruhan) → direduksi menjadi 100 kasus.
  - 100 kasus dikelompokkan ke dalam 5 data uji masing-masing berisi 20 kasus.
  - Dilakukan sebanyak 10 kali pada masing-masing metode penyusunan formula, yaitu dengan penggunaan algoritma *Berlekamp-Massey* dan interpolasi *Lagrange*.

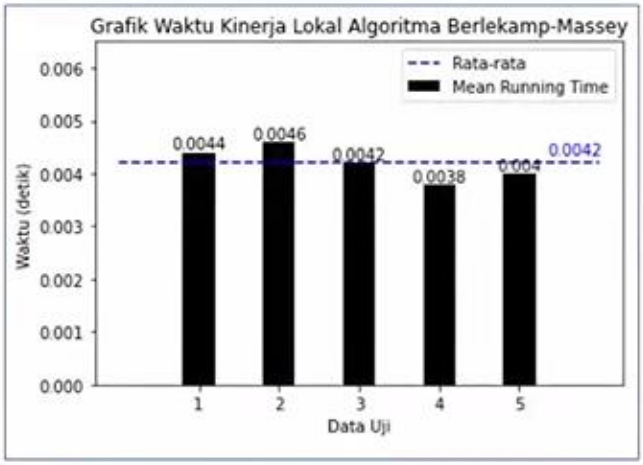
Dengan metode tersebut akan menghasilkan hasil sebgai berikut

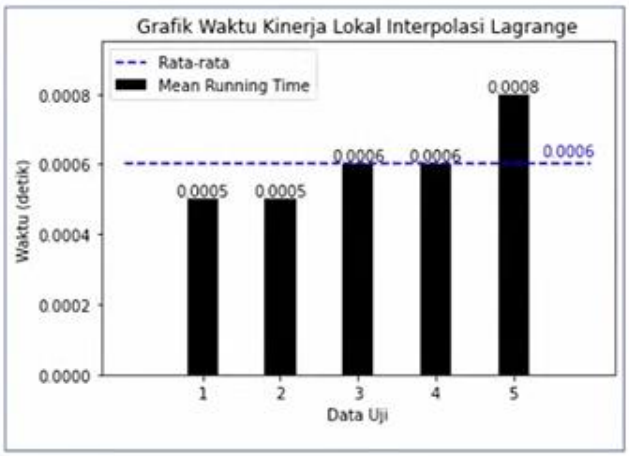


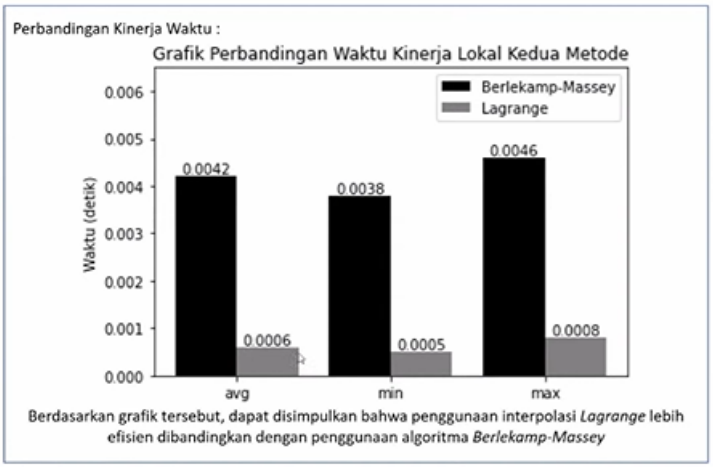




Ditemukan hasil dari kedua metode tidak terdapat perbedaan yang terlalu signifikan oleh karena itu kita lakukan uji coba local yang akan menghasilkan hasil sebagai berikut




Dari data tersebut dapat diambil analasis sebagai berikut

| Faktor Pembanding | Algoritma Berlekamp - massey | Interpolasi Lagrange |
|---|---|---|

| | | |
|---|---|---|
| Kelebihan | Pemilihan barisan bilangan berpola yang dipakai sebagai masukan relative lebih mudah | Menghasilkan keluaran formula polynomial yang relative lebih cepat daripada liniear rekuren homogen |
| Kekurangan | Menghasilkan keluaran linear rekuren homogen yang relative lebih lambat daripada formula polinomial | Pemilihan barisan bilangan berpola yang dipakai sebagai masukan relative lebih sulit |

Jadi dapat disimpulkan bahwa kedua metode bisa digunakan untuk menyelesaikan soal Battle Of The Bastards – Kings vs Rooks, akan tetapi metode interpolasi lagrange lebih cepat karena menggunakan lebih sedikit iterasi.