REINFORCEMENT LEARNING
"AN RL AGENT THAT CAN ACTUALLY PLAY 2048"

Dibuat Oleh Kelompok 1:

Nanditha Nabiilah Putri (G1A021001)

Siti Zubaidah (G1A021002)

Elisa (G1A021008)

Dosen Mata Kuliah:

Arie Vatresia, S.T., M.T.I., Ph.D

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

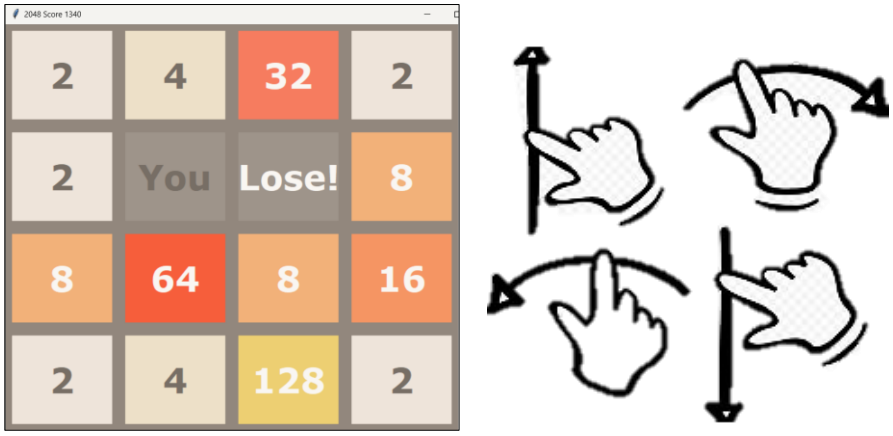
UNIVERSITAS BENGKULU

2024

DAFTAR ISI

CARA KERJA 2048

Game yang ada di dalam file ini merupakan implementasi dari permainan 2048, sebuah game puzzle yang cukup populer. Dalam permainan 2048, pemain menggerakkan ubin-ubin bernilai angka kelipatan dua (2, 4, 8, 16, dst.) di atas papan berukuran 4x4. Tujuan permainan ini adalah untuk menggabungkan ubin-ubin yang memiliki nilai yang sama hingga mencapai angka 2048.



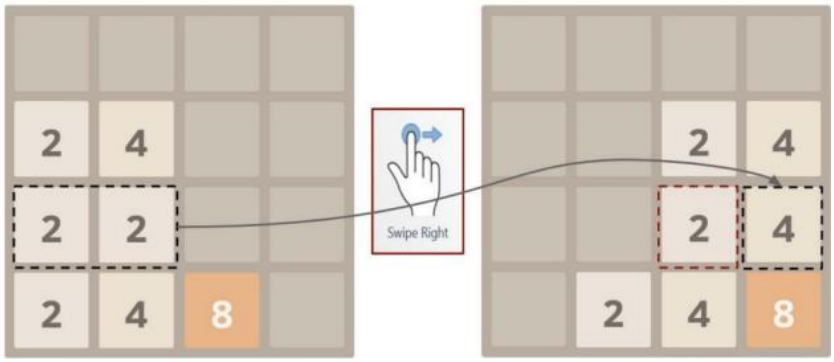
Gambar 1 Permainan 2048

A. Kisi:

1. Ukuran PapanGrid adalah 4x4 dengan 16 kotak.
2. Pemain bisa menggeser semua ubin ke arah atas, bawah, kiri, atau kanan.
3. Penggabungan ubin dengan nilai yang sama akan bergabung menjadi satu ubin dengan nilai yang merupakan penjumlahan keduanya.
4. Muncul ubin baru setelah setiap gerakan, ubin baru dengan nilai 2 atau 4 muncul secara acak di kotak kosong.
5. Tujuan permainan ini adalah mencapai ubin dengan nilai 2048.
6. Permainan berakhir ketika tidak ada lagi gerakan yang dapat dilakukan karena papan penuh dan tidak ada ubin yang bisa digabungkan.
7. Game ini mengharuskan pemain untuk memikirkan langkah-langkah ke depan agar dapat menggabungkan ubin seefisien mungkin, sekaligus menjaga ruang kosong agar tidak cepat terisi penuh.

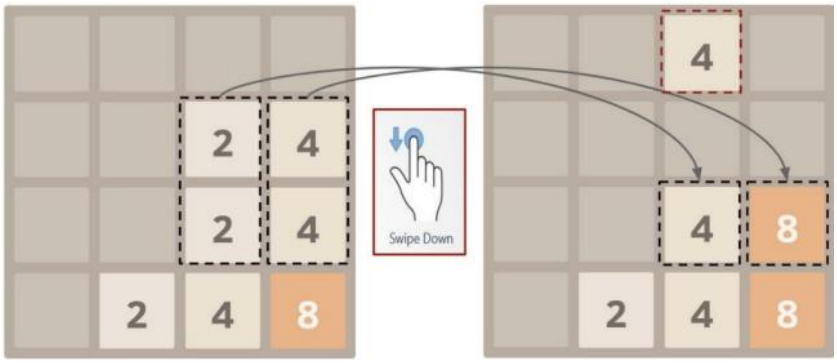
B. Aturan:

1. Jika kita memilih untuk menggeser ke kanan, semua kotak pada grid akan bergeser sejauh mungkin ke arah kanan. Selain itu, jika dua angka yang sama bertemu selama pergeseran, keduanya akan digabungkan menjadi satu dengan cara dijumlahkan. Setelah itu, angka baru akan muncul secara acak di salah satu sel kosong dalam grid, dan angka tersebut selalu berupa 2 atau 4.



Gambar 2 Pergerakan ubin secara horizontal

2. Jika kita menggeser ke bawah, semua kotak dalam grid akan bergerak sejauh mungkin ke arah bawah. Jika ada dua angka yang sama bertemu selama pergeseran, mereka akan digabungkan. Setelah itu, angka baru akan muncul secara acak di salah satu sel kosong, seperti sebelumnya.



Gambar 3 Pergerakan ubin secara vertikal

3. Permainan akan terus berlanjut hingga kita kehabisan langkah, atau saat petak terbesar di papan adalah 2048. Banyak permainan yang memungkinkan kita untuk terus bermain setelah mendapatkan 2048, dan juga akan menganggap hal yang sama untuk permainan yang akan dimainkan oleh pemain.

RINGKASAN

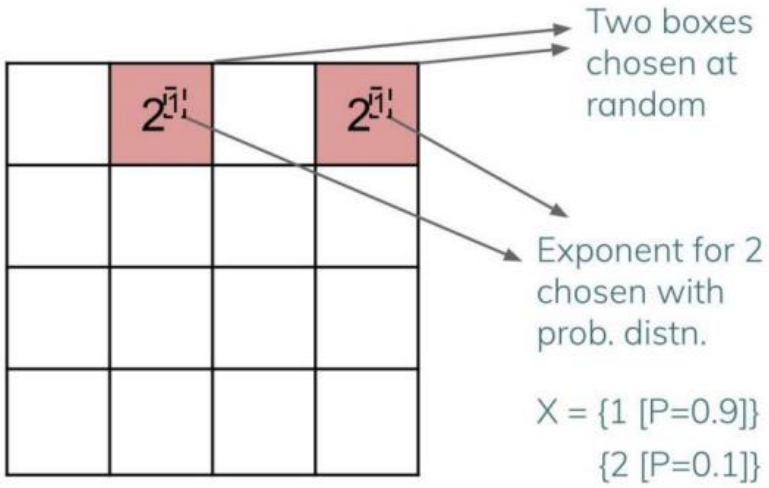
A. Batasan

Batasan Lingkungan Reinforcement Learning untuk masalah ini disusun sebagai berikut:

- Papan permainan sebagai lingkungan
- Agen yang berperan adalah agen RL yang telah dilatih.
- Status didefinisikan oleh angka-angka yang ada di papan permainan.
- Aksi yang dapat diambil adalah bergerak ke kiri, atas, kanan, atau bawah.
- Hadiah didasarkan pada total nilai dari angka yang berhasil digabungkan.

B. Detail Implementasi

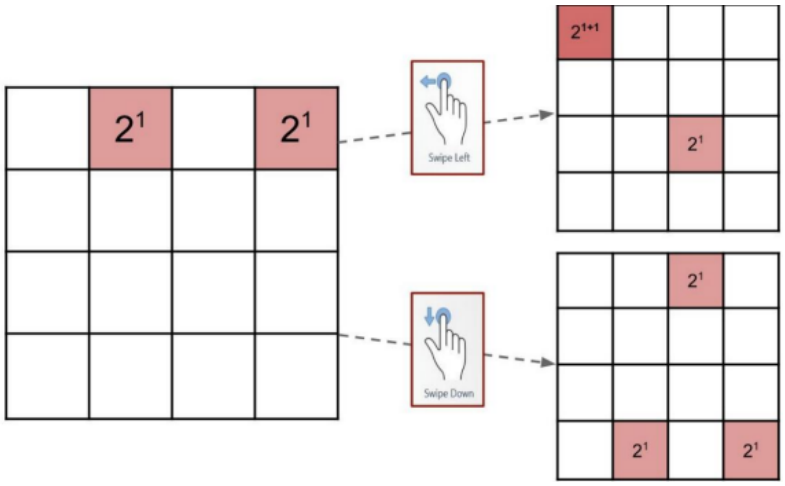
- Untuk menginisialisasi papan, kami memilih dua petak secara acak untuk diisi. Untuk dua petak ini, kami memilih x menjadi 1 dengan probabilitas 0,9 dan 2 dengan probabilitas 0,1, dan kemudian, 2 x menjadi nilai petak-petak ini.



Gambar 4 Inisialisasi papan

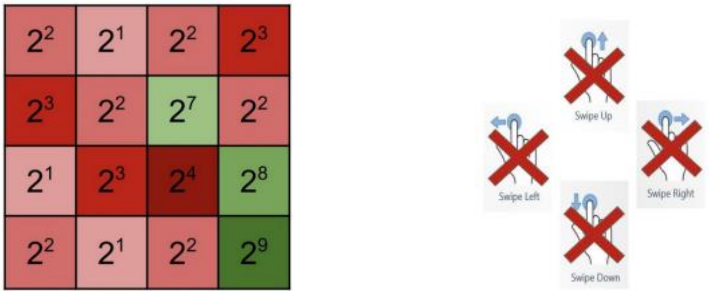
- Selanjutnya, tergantung pada tindakan pengguna seperti gambar 5, kami memperbarui nilai x untuk setiap petak di kisi. Jika terjadi penggabungan, kami menambahkan nilai x, seperti yang terjadi pada kasus di atas. Selain itu, kami akan membuat nilai x acak untuk salah satu sel kosong lagi, dalam hal

yang mirip dengan inisialisasi papan, dengan satu-satunya perbedaan adalah kali ini, kami hanya membuat angka untuk 1 petak, bukan 2.



Gambar 5 Tindakan

- Permainan berakhir saat Anda mencapai situasi di mana Anda kehabisan gerakan yang tersedia. Misalnya, dalam kasus di atas, Anda tidak dapat menggeser ke kiri, atas, bawah, atau kanan. Oleh karena itu, permainan berakhir.

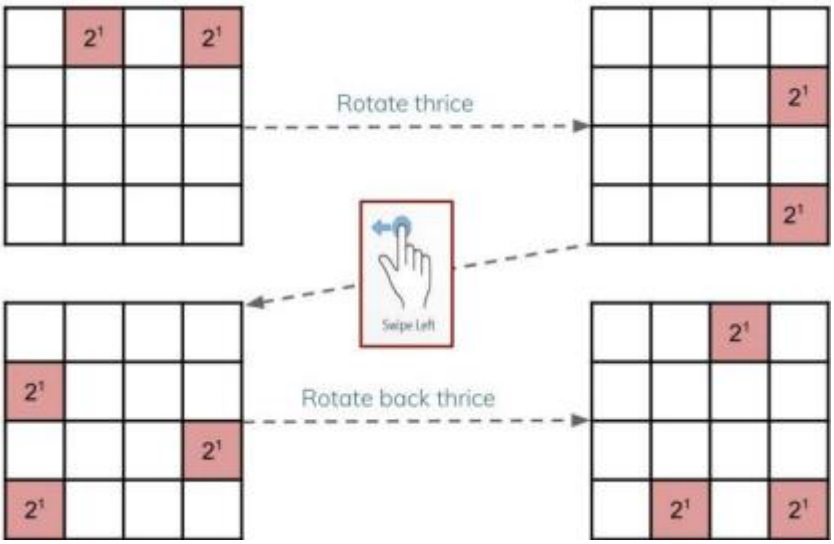


Gambar 6 Game berhenti

1. Memeriksa apakah tindakan geser kiri tersedia atau tidak.

Gambar 7 Pemeriksaan

2. Memutar matriks sekali, lalu memeriksa apakah tindakan geser kiri tersedia atau tidak, yang akan memeriksa apakah tindakan geser atas tersedia.
3. Memutar lagi sekali dan memeriksa geser kiri, ini akan memeriksa tindakan geser kanan.
4. Memutar sekali lagi dan memeriksa geser kiri, yang membantu kita memeriksa apakah tindakan geser bawah tersedia atau tidak.
5. Pendekatan ini membantu menjaga kode tetap modular. Selain itu, kita hanya perlu memahami logika untuk memeriksa satu jenis tindakan (geser kiri), yang membuat fungsi lebih mudah untuk ditulis.
6. Dengan cara serupa, kita dapat mengeksekusi semua tindakan hanya menggunakan kode untuk geser kiri. Sebagai contoh, jika kita ingin melakukan geser ke bawah, langkah-langkahnya adalah:
7. Putar matriks tiga kali dengan sudut 90 derajat.



Gambar 8 Pengecekan 90 derajat

8. Eksekusi geser kiri.
9. Putar kembali matriks tiga kali dengan sudut 90 derajat.
10. Dengan logika yang sama, kita bisa mengeksekusi semua tindakan yang memungkinkan.

PERBANDINGAN PENDEKATAN PEMBELAJARAN

A. Sucked

Approach	Mean Score	Max Score
Random	1093	2736
Q-Learning	1181	3324
DDQN	1205	3530
Human Level	14321	20214

Gambar 10 Analisis pendekatan buruk

Q-Learning tidak berjalan dengan baik karena banyaknya pasangan keadaan-tindakan dalam lingkungan. Ini menghambat kemampuan model untuk mempelajari kebijakan yang sederhana dan dapat digeneralisasikan.

Selain itu, DDQN menderita masalah miopia, dalam arti bahwa ia mengetahui tindakan terbaik dalam jangka pendek, tetapi tidak memiliki pemahaman tentang nilainya dalam jangka panjang. Karena ini adalah permainan strategi, pandangan jangka panjang sebenarnya sangat penting.

B. Didn't Suck

Approach	Mean Score	Max Score
Random	1093	2736
Q-Learning	1181	3324
DDQN	1205	3530
MC (1-step)	1811	6192
MC (2-step)	7648	16132
MC (3-step)	8609	16248
Human Level	14321	20214

Gambar 11 Analisis pendekatan baik

1. Metode yang akhirnya berhasil untuk kami sebenarnya cukup sederhana, yaitu dengan menggunakan Pencarian Pohon Monte Carlo (Monte Carlo Tree Search).
2. Cara kerja dasar yang kemungkinan langkah yang tersedia dan menentukan nilai terkait dari setiap langkah. Langkah berikutnya yang kami pilih adalah langkah dengan nilai tertinggi.
3. Menerapkan pendekatan yang lebih rumit daripada hanya melihat langkah berikutnya. Misalnya, kami bisa memeriksa kombinasi beberapa langkah ke depan, katakanlah tiga langkah. Jadi, kami mengevaluasi setiap kombinasi langkah yang mungkin terjadi untuk tiga putaran ke depan, lalu memilih langkah terbaik berdasarkan nilai keseluruhan dari kombinasi tersebut.
4. Pendekatan ini sangat efektif, dan alasannya cukup jelas: pada setiap titik waktu, kami membuat keputusan tentang langkah masa depan dengan mempertimbangkan strategi jangka panjang. Hal ini, tentu saja, memberikan hasil yang lebih baik.
5. Untuk mendukung pendapat kami, kami menyajikan kembali skor rata-rata dan skor maksimal dari setiap strategi, kali ini menyertakan hasil dari Pendekatan Monte Carlo. Meskipun pendekatan Monte Carlo satu langkah tidak memberikan hasil yang baik, pendekatan Monte Carlo multi-langkah kami berhasil melampaui kinerja tingkat manusia.