
INSTITUT TEKNOLOGI BANDUNG
1920

## Modul 3: Beyond Classical Search

# Genetic Algorithm

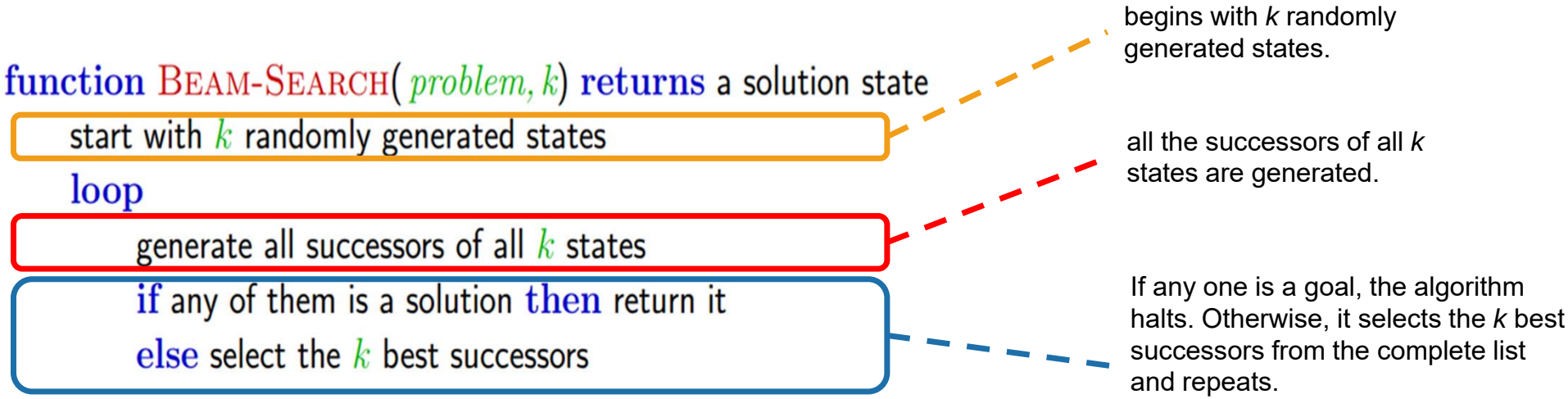
KK IF – Teknik Informatika – STEI ITB

Inteligensi Buatan
(*Artificial Intelligence*)



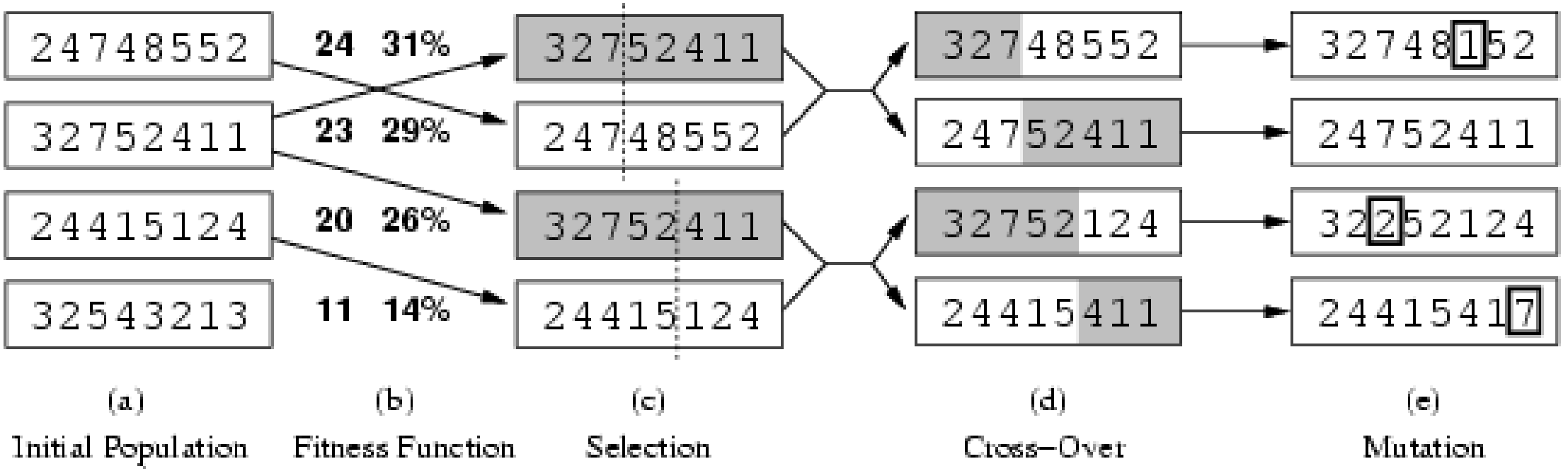
# Genetic Algorithm

# Local Beam Search




In a local beam search, useful information is passed among the parallel search k threads

# Genetic Algorithm: Illustration (Russel & Norvig, 2010)

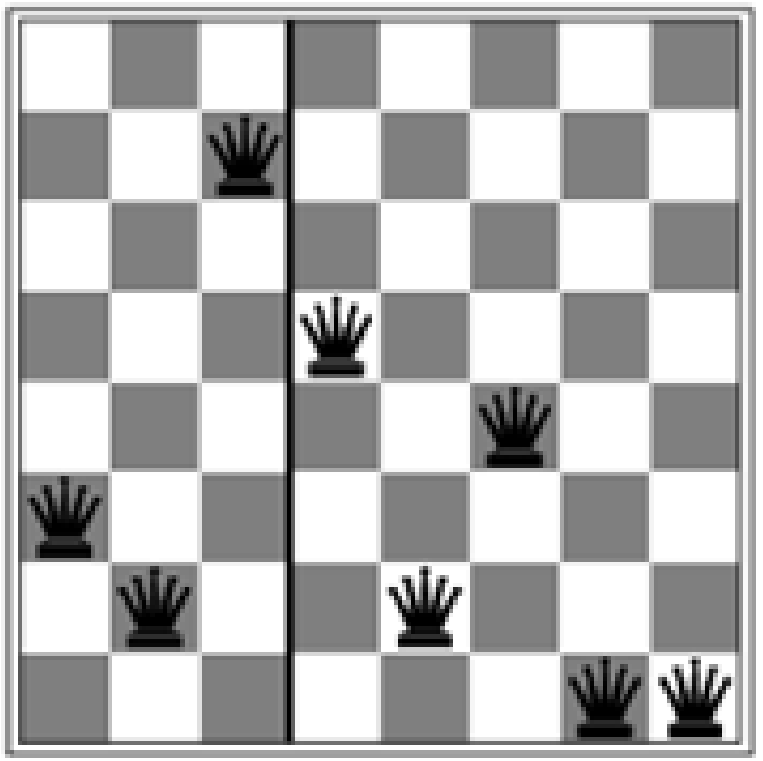


# State / Individual

A state is represented as a string over a finite alphabet (often a string of 0s and 1s)

One character ~ one variable

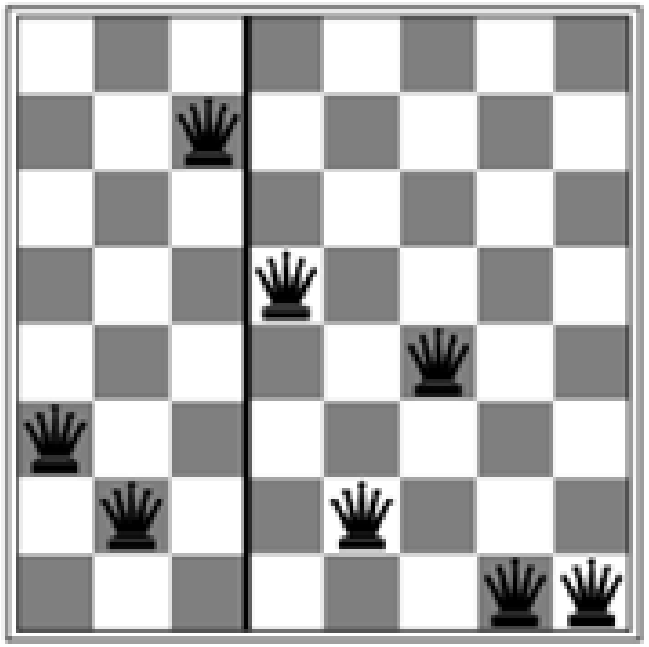
32752411



# State Value: Fitness Function

- Evaluation function (fitness function).
- Higher values for better states.
- Fitness function for n-queen: number of non-attacking pairs of queens
  - min = 0
  - max = (8 × 7)/2 = 28 (global maximum)

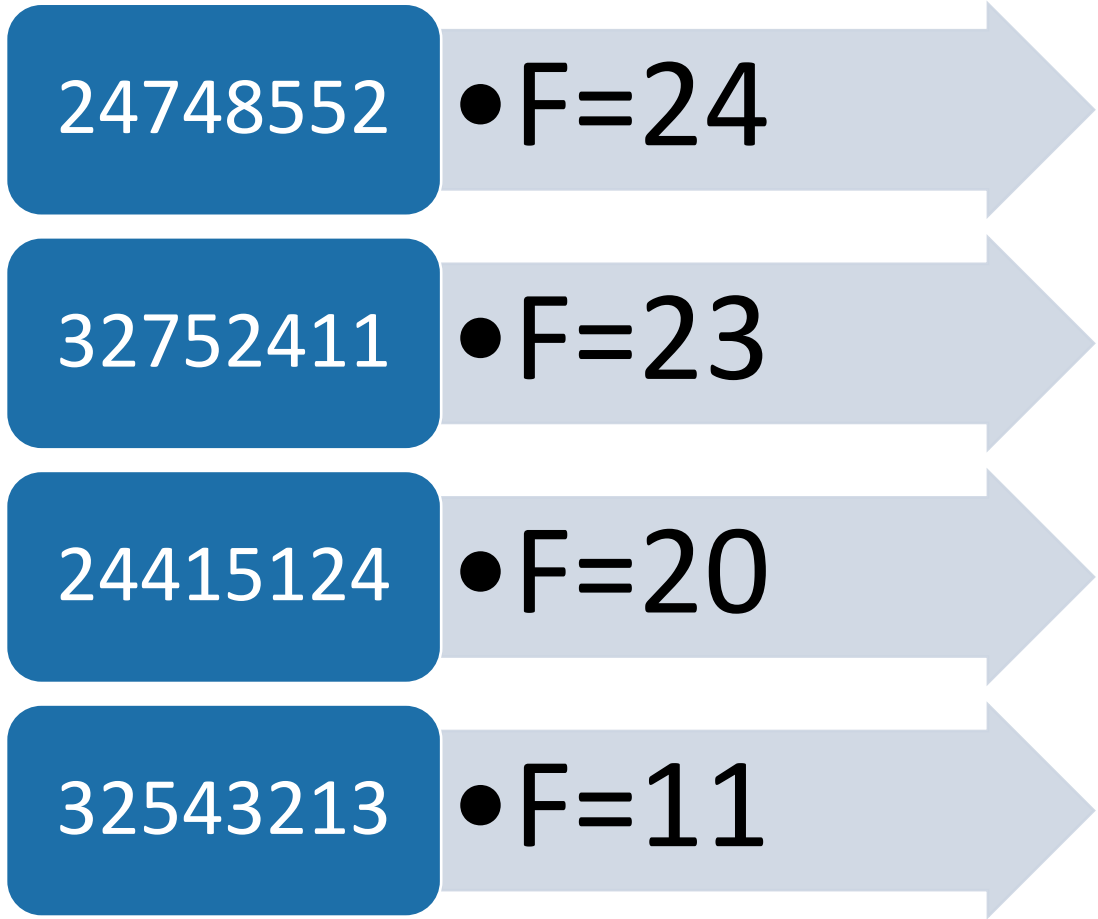
32752411



F(32752411)=28-5=23

# Initial Population & Successor Function

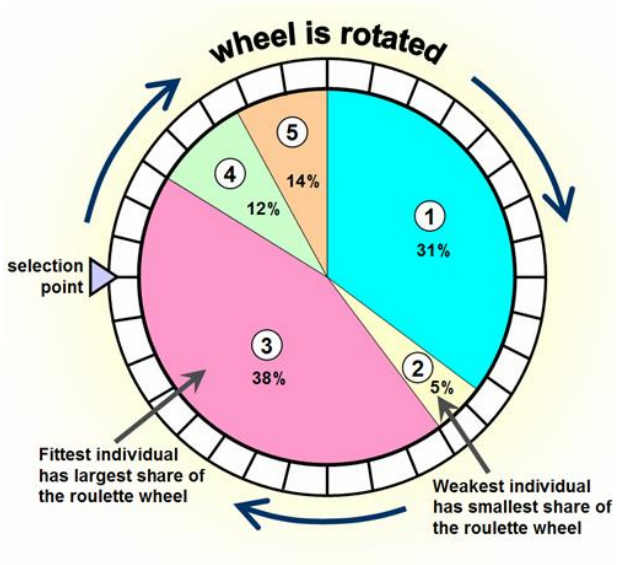
- GA starts with *k* randomly generated states.
- A successor state is generated by combining two parent states
- Produce the next generation of states by selection, crossover, and mutation

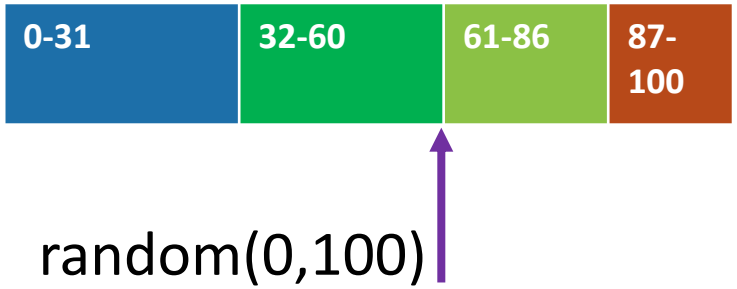


# Random Selection of Parent States

## Probability of random selection

| | |
|---|---|
| 24748552 | • 24/(24+23+20+11)<br>• = 31% |
| 32752411 | • 23/(24+23+20+11)<br>• =29% |
| 24415124 | • 20/(24+23+20+11)<br>• =26% |
| 32543213 | • 11/(24+23+20+11)<br>• =14% |




32752411

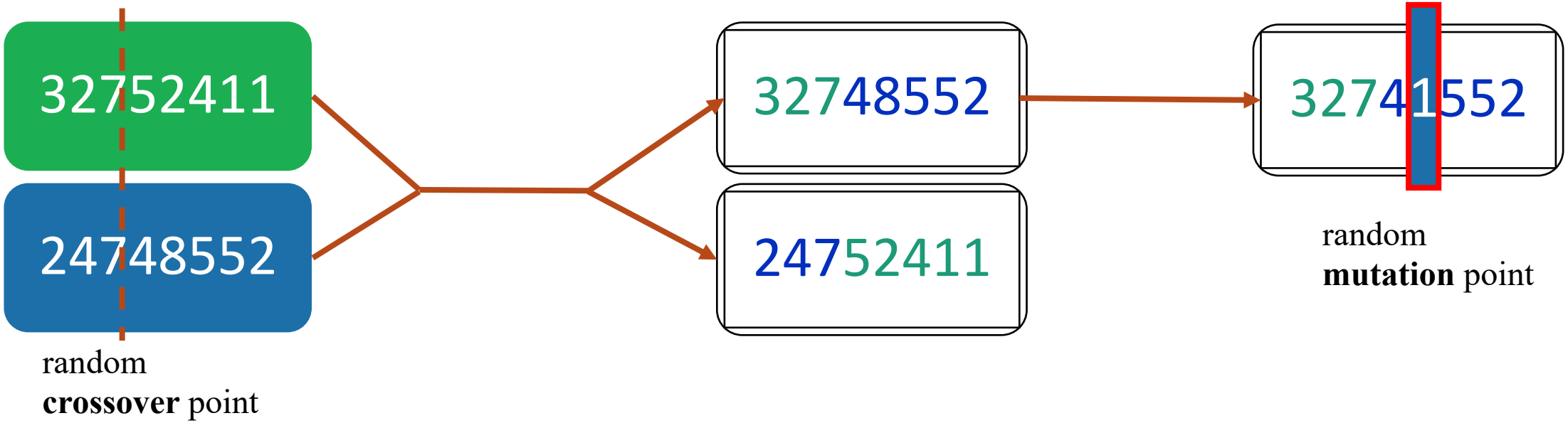
24748552

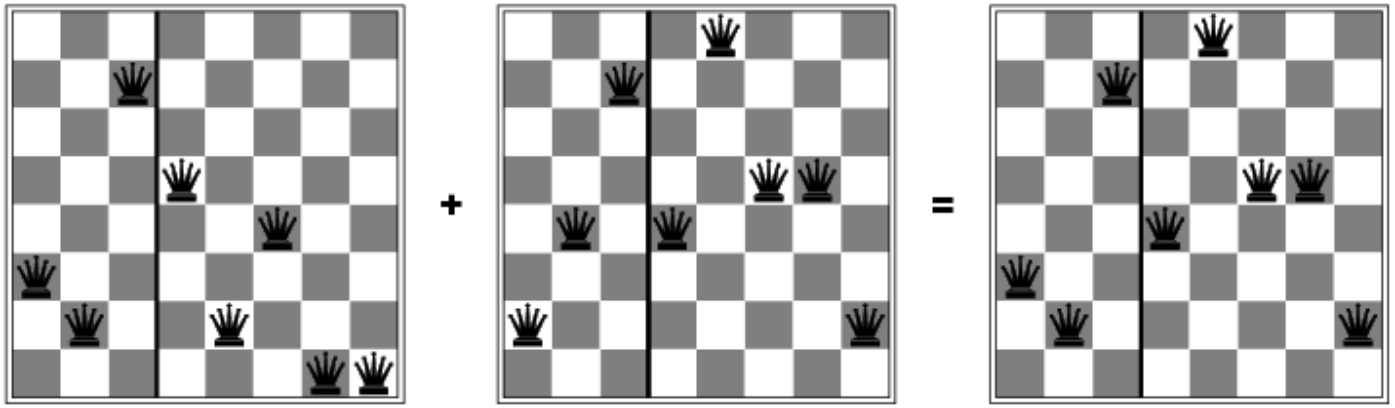
32752411

24415124

EDUNEX ITB

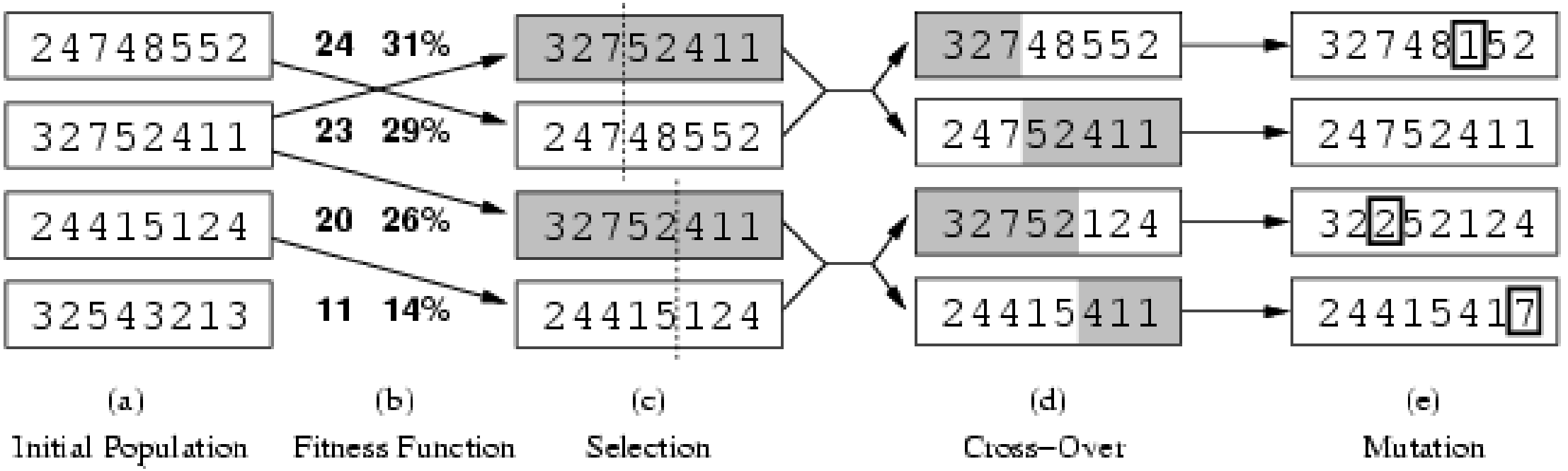
# Cross Over / Reproduce & Mutation




In more popular version, each mating of two parents produces only one offspring, not two (Russel & Norvig, 2010)

# Genetic Algorithm: Illustration (Russel & Norvig, 2010)



# Genetic Algorithm

**function** GENETIC-ALGORITHM(*population*, FITNESS-FN) **returns** an individual
  **inputs**: *population*, a set of individuals
    FITNESS-FN, a function that measures the fitness of an individual

  **repeat**
    *new_population* ← empty set
    **for** *i* = 1 **to** SIZE(*population*) **do**
      *x* ← RANDOM-SELECTION(*population*, FITNESS-FN)
      *y* ← RANDOM-SELECTION(*population*, FITNESS-FN)
      *child* ← REPRODUCE(*x*, *y*)
      **if** (small random probability) **then** *child* ← MUTATE(*child*)
      add *child* to *new_population*
    *population* ← *new_population*
  **until** some individual is fit enough, or enough time has elapsed
  **return** the best individual in *population*, according to FITNESS-FN



# Summary

*k* randomly generated states (population: k individual)

Fitness function (state value)

successor function: combining two parent states (selection, cross-over, mutation)



# Terima Kasih




INSTITUT TEKNOLOGI BANDUNG
1920