# Алгоритмы и структуры данных

Лекция 9. Эвристические алгоритмы поиска.

(c) Глухих Михаил Игоревич, glukhikh@mail.ru

# Эвристические алгоритмы

- Семейство «мета-алгоритмов»

# Эвристические алгоритмы

- Семейство «мета-алгоритмов»
- Предназначены в первую очередь для задач оптимизации, которые точно решаются «долго» -- $O(e^n)$, $O(n!)$, …

# Эвристические алгоритмы

- Семейство «мета-алгоритмов»
- Предназначены в первую очередь для задач оптимизации, которые точно решаются «долго» -- $O(e^n)$, $O(n!)$, …
- Обеспечивают приближённое решение за приемлемое время

# Эвристические алгоритмы

- Семейство «мета-алгоритмов»
- Предназначены в первую очередь для задач оптимизации, которые точно решаются «долго» -- $O(e^n)$, $O(n!)$, …
- Обеспечивают приближённое решение за приемлемое время
- Содержат элемент случайности

# Эвристические алгоритмы

- Семейство «мета-алгоритмов»
- Предназначены в первую очередь для задач оптимизации, которые точно решаются «долго» -- $O(e^n)$, $O(n!)$, …
- Обеспечивают приближённое решение за приемлемое время
- Содержат элемент случайности
- Часто имитируют какие-либо естественные процессы

- Близкое понятие: мягкие вычисления (см. Википедия)

# Полностью случайный алгоритм

- Сгенерировать N случайных решений
- Выбрать из них лучшее
- Сказать, что это и есть «хорошее приближение»

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи
- «Хромосома» – возможное решение (любое)

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- «Хромосома» – возможное решение (любое)
- «Популяция» – набор хромосом = набор возможных решений

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- «Хромосома» – возможное решение (любое)
- «Популяция» – набор хромосом = набор возможных решений
- Функция отбора: Хромосома → Число (чем больше, тем соответствующее решение лучше)

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- «Хромосома» – возможное решение (любое)
- «Популяция» – набор хромосом = набор возможных решений
- Функция отбора: Хромосома → Число (чем больше, тем соответствующее решение лучше)
- Функция скрещивания: Хромосома, Хромосома → Хромосома (формирует из двух решений одно «смешанное»)

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- «Хромосома» – возможное решение (любое)
- «Популяция» – набор хромосом = набор возможных решений
- Функция отбора: Хромосома → Число (чем больше, тем соответствующее решение лучше)
- Функция скрещивания: Хромосома, Хромосома → Хромосома (формирует из двух решений одно «смешанное»)
- Функция мутации: Хромосома → Хромосома (модифицирует решение)

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- Алгоритм:
  - Генерируем популяцию размера N

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- Алгоритм:
  - Генерируем популяцию размера N
  - Отбор: выкидываем из неё Nw худших (например 50%)

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- Алгоритм:
  - Генерируем популяцию размера N
  - Отбор: выкидываем из неё Nw худших (например 50%)
  - Скрещивание: формируем из оставшихся N-Nw хромосом Nw новых, применяя функцию скрещивания

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- Алгоритм:
  - Генерируем популяцию размера N
  - Отбор: выкидываем из неё Nw худших (например 50%)
  - Скрещивание: формируем из оставшихся N-Nw хромосом Nw новых, применяя функцию скрещивания
  - Мутация: Nm из полученных N хромосом (например 10%)

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- Алгоритм:
  - Генерируем популяцию размера N
  - Отбор: выкидываем из неё Nw худших (например 50%)
  - Скрещивание: формируем из оставшихся N-Nw хромосом Nw новых, применяя функцию скрещивания
  - Мутация: Nm из полученных N хромосом (например 10%)
  - Получаем популяцию следующего поколения и возвращаемся к этапу отбора

# Генетический алгоритм

- Один из видов «эвристических» алгоритмов, обеспечивающих приближённое решение задачи

- Алгоритм:
  - Генерируем популяцию размера N
  - Отбор: выкидываем из неё Nw худших (например 50%)
  - Скрещивание: формируем из оставшихся N-Nw хромосом Nw новых, применяя функцию скрещивания
  - Мутация: Nm из полученных N хромосом (например 10%)
  - Получаем популяцию следующего поколения и возвращаемся к этапу отбора
  - Сформировав таким образом сколько-то поколений, берём лучшее из имеющихся в популяции решений

# Задача о ранце

- Есть рюкзак грузоподъёмностью L
- Есть набор из M предметов ценностью Ci и весом Wi
- Необходимо выбрать, какие из них брать:
  Sum(Wi) <= L, Sum(Ci) → Max

# Задача о ранце -- варианты

- Есть рюкзак грузоподъёмностью L
- Есть набор из M предметов ценностью Ci и весом Wi
- Необходимо выбрать, какие из них брать:
  Sum(Wi) <= L, Sum(Ci) → Max

- Задача 0/1 – каждый предмет либо берётся, либо нет

# 0/1 задача о ранце – генетический алгоритм

- Хромосома = набор бит вида 011010001 (кладём в рюкзак 2-й, 3-й, 5-й и 9-й предметы)

# 0/1 задача о ранце – генетический алгоритм

- Хромосома = набор бит вида 011010001 (кладём в рюкзак 2-й, 3-й, 5-й и 9-й предметы)
- Скрещивание

```
011010001
101010100
---------
RR1010R0R – все R случайны
```

# 0/1 задача о ранце – генетический алгоритм

- Хромосома = набор бит вида 011010001 (кладём в рюкзак 2-й, 3-й, 5-й и 9-й предметы)
- Скрещивание

```
011010001
101010100
---------
RR1010R0R – все R случайны
```

- Мутация – случайное изменение одного или нескольких бит

# 0/1 задача о ранце – генетический алгоритм

- Хромосома = набор бит вида 011010001 (кладём в рюкзак 2-й, 3-й, 5-й и 9-й предметы)
- Скрещивание

```
011010001
101010100
---------
RR1010R0R – все R случайны
```

- Мутация – случайное изменение одного или нескольких бит
- Функция отбора
  - Равна суммарной стоимости предметов, если влезли
  - Если не влезли – например, избыток веса со знаком минус

# Задача коммивояжёра – генетический алгоритм

- Обойти все вершины графа по одному разу и вернуться в исходное с наименьшими затратами (путь наименьшей длины)

- Генерация популяции – случайная перестановка чисел от 0 до N-1

# Случайная перестановка

- Collections.shuffle(mutableList)

- Дано: List<T> / Array<T> inputList размера N с уникальными элементами, то есть inputList[i] != inputList[j]
- Требуется: сформировать outputList того же размера, в котором элементы inputList переставлены так, чтобы...

# Случайная перестановка

- Collections.shuffle(mutableList)

- Дано: List<T> / Array<T> inputList размера N с уникальными элементами, то есть inputList[i] != inputList[j]
- Требуется: сформировать outputList того же размера, в котором элементы inputList переставлены так, чтобы… Все N! перестановок были равновероятны

# Случайная перестановка – варианты решения

# Случайная перестановка – варианты решения

- Много раз поменять местами случайно выбранную пару элементов

# Случайная перестановка – варианты решения

- Много раз поменять местами случайно выбранную пару элементов (ПЛОХО)
- Сформировать рядом список случайных чисел $1…N^3$ и отсортировать исходный в соответствии с порядком данного случайного списка

# Случайная перестановка – варианты решения

- Много раз поменять местами случайно выбранную пару элементов (ПЛОХО)
- Сформировать рядом список случайных чисел $1…N^3$ и отсортировать исходный в соответствии с порядком данного случайного списка (ДОРОГО)
- Перестановка на месте – для i от 0 до N-1 поменять местами элемент i и элемент j, где j случайное число от i до N-1 – [Кормен], глава 5

# Задача коммивояжёра – генетический алгоритм

- Обойти все вершины графа по одному разу и вернуться в исходное с наименьшими затратами (путь наименьшей длины)

- Генерация популяции – случайная перестановка чисел от 0 до N-1

# Задача коммивояжёра – генетический алгоритм

- Обойти все вершины графа по одному разу и вернуться в исходное с наименьшими затратами (путь наименьшей длины)

- Генерация популяции – случайная перестановка чисел от 0 до N-1
- Мутация – перестановка местами случайной пары

# Задача коммивояжёра – генетический алгоритм

- Обойти все вершины графа по одному разу и вернуться в исходное с наименьшими затратами (путь наименьшей длины)

- Генерация популяции – случайная перестановка чисел от 0 до N-1
- Мутация – перестановка местами случайной пары
- Скрещивание -- ??? См. lesson8.genetic

# Алгоритм имитации отжига

- Имитирует процесс остывания полужидкого металла с образованием кристаллической решётки
- В начале процесса (пока металл не застыл) допустимы большие отклонения
- В конце процесса допустимы только маленькие отклонения

# Алгоритм имитации отжига

- Параметры
  - T(i) – зависимость температуры от номера итерации (монотонно убывает)
  - T(1) = Tstart
  - Tfinish (или число итераций)

# Алгоритм имитации отжига

- Параметры
  - T(i) – зависимость температуры от номера итерации (монотонно убывает)
  - T(1) = Tstart
  - Tfinish (или число итераций)
  - Функция генерации случайного решения
  - Функция мутации

# Алгоритм имитации отжига

- Параметры
  - T(i) – зависимость температуры от номера итерации (монотонно убывает), T(1) = Tstart, Tfinish (или число итераций)
  - Функция генерации случайного решения
  - Функция мутации
- Ход решения
  - Сгенерировать случайное решение
  - В цикле: i++, мутация решения, сравнение с предыдущим

# Алгоритм имитации отжига

- Параметры
  - T(i) – зависимость температуры от номера итерации (монотонно убывает), T(1) = Tstart, Tfinish (или число итераций)
  - Функция генерации случайного решения
  - Функция мутации
- Ход решения
  - Сгенерировать случайное решение
  - В цикле: i++, мутация решения, сравнение с предыдущим
    - Если новое решение лучше – берём его

# Алгоритм имитации отжига

- Параметры
  - T(i) – зависимость температуры от номера итерации (монотонно убывает), T(1) = Tstart, Tfinish (или число итераций)
  - Функция генерации случайного решения
  - Функция мутации
- Ход решения
  - Сгенерировать случайное решение
  - В цикле: i++, мутация решения, сравнение с предыдущим
    - Если новое решение лучше – берём его
    - Если лучше старое – берём новое с вероятностью P(dE/T)

# Алгоритм имитации отжига

- Параметры
  - T(i) – зависимость температуры от номера итерации (монотонно убывает), T(1) = Tstart, Tfinish (или число итераций)
  - Функция генерации случайного решения
  - Функция мутации
- Ход решения
  - Сгенерировать случайное решение
  - В цикле: i++, мутация решения, сравнение с предыдущим
    - Если новое решение лучше – берём его
    - Если лучше старое – берём новое с вероятностью P(dE/T)
    - dE = разница в оценке решений

# Алгоритм имитации отжига

- Параметры
  - T(i) – зависимость температуры от номера итерации (монотонно убывает), T(1) = Tstart, Tfinish (или число итераций)
  - Функция генерации случайного решения
  - Функция мутации
- Ход решения
  - Сгенерировать случайное решение
  - В цикле: i++, мутация решения, сравнение с предыдущим
    - Если новое решение лучше – берём его
    - Если лучше старое – берём новое с вероятностью P(dE/T)
    - P(0)=1, P(inf)=0, P монотонно убывает, e.g. exp(-x)

# Алгоритм имитации отжига

- Пример см. lesson8.annealing – задача коммивояжёра
- Мутация – «переворачиваем» случайно выбранную часть списка

# Алгоритм колонии муравьёв

- В первую очередь ориентирован на задачи поиска сложных путей на графе
- Имитирует поведение колонии муравьёв при поиске пути к некоторой цели

# Алгоритм колонии муравьёв

- В первую очередь ориентирован на задачи поиска сложных путей на графе
- Имитирует поведение колонии муравьёв при поиске пути к некоторой цели
  - Каждый муравей ищет путь по отдельности условно-случайным образом, оставляя след из «феромонов»
  - Последующие муравьи с большей вероятностью пойдут туда, где феромонов оставлено больше

# Алгоритм колонии муравьёв

- Феромон
  - Размещается на рёбрах
  - Уровень – некоторое вещественное число
  - Со временем испаряется (умножается на константу <1)
  - Прошедший муравей добавляет слагаемое, обратно пропорциональное длине ребра

# Алгоритм колонии муравьёв

- Муравей
  - Вначале располагается в выбранной стартовой или в случайной вершине, в зависимости от задачи
  - На каждом шаге выбирает ребро
    - Вес ребра для муравья зависит от его длины (чем она больше, тем вес меньше) и от уровня феромона на ребре (чем он больше, тем вес больше)
    - Конкретное ребро выбирается случайным образом, ребро с большим весом выбирается чаще

# Задание по эвристическим алгоритмам (урок 8)

- Решить 0/1 задачу о ранце одним из трёх методов
  - Генетический алгоритм
  - Алгоритм имитации отжига
  - Алгоритм колонии муравьёв
- Или – решить задачу коммивояжёра с помощью алгоритма колонии муравьёв
- Или – решить одну из этих задач каким-либо ещё эвристическим алгоритмом (NB: полностью случайный со слайда 7 не допускается!)
- Дедлайн всех уроков: 10 декабря 23:59

# Итоги

- Рассмотрели
  - Эвристические алгоритмы
    - Генетический алгоритм
    - Алгоритм имитации отжига
    - Алгоритм колонии муравьёв
- Далее
  - P versus NP