# Алгоритмы и структуры данных

## Лекция 27

## Решение неполиномиальных задач.

Сергей Леонидович Бабичев

# NP-задачи

- Почти все изученные алгоритмы имели небольшую, полиномиальную сложность.
- Нам встречались алгоритмы, решающиеся неполиномиально — задача о рюкзаке, задача коммивояжёра.
- Стоит ли искать полиномиальные решение таких задач?
- Или такой поиск принципиально невозможен и мы будем искать вечный двигатель?
- Или наоборот, пытаясь безуспешно доказать неполиномиальность решения, мы найдём полиномиальное?

# Принцип сведения

- *Задача* — некий запрос, имеющий входные параметры и условия проверки правильности ответа.
- *Экземпляр задачи* — задача с заданным входом.
- Пример общей задачи: сформулированная с общей точки зрения задача о рюкзаке.
- Пример экземпляра задачи: задача о рюкзаке с конкретно заданными входными данными.
- Решение конкретного экземпляра задачи может оказаться лёгким.
- Требуется, чтобы данный алгоритм решал *всевозможные* экземпляры задач.

# Принцип сведения

- Пусть имеются задачи $P_1$ и $P_2$ и имеется алгоритм решения задачи $P_1(D)$ через $P_2$:
  1. Преобразуем $D$ в экземпляр $F$ задачи $P_2$ (проводим *редукцию*).
  2. Решаем $P_2(F)$
  3. Результат возвращаем как ответ на $P_1(D)$.
- Если преобразование задач корректно, то преобразование даст верный результат, то есть $P_1(D) = P_2(F)$.
- Пусть сложность сведения $D \to F$ есть $O(f(n))$.
- Если $T_2$ имеет сложность $O(f'(n))$, то совокупное время решения $P_1$ есть $O(f(n) + f'(n))$.
- Если известно, что нижний предел сложности решения $T_1$ есть $\Omega(f'(n))$, то нижний предел сложности $T_2$ не может быть меньше $\Omega(f'(n) - f(n))$.

# Задача разрешимости

- При сведении задач сделаем, чтобы ответы на все экземпляры будут идентичными.
- Задачи *разрешимости* отвечают *Да* или *Нет* на любые запросы.
- Например, задачу рюкзака сводим к задаче разрешимости так: имеется набор предметов с их стоимостью и ценами а также ёмкость рюкзака и некоторая сумма $S$. Выходом алгоритмы должно быть логическое значение: можно ли обеспечить нужную стоимость предметов в рюкзаке не меньшую, чем $S$.

# Сведение алгоритмов: поиск ближайшей пары чисел

**Задача 1.** Найти в множестве чисел пару с наименьшей разностью.

- Задача сводится к задаче разрешимости: для данного входа и числа $t$ существует ли пара $i, j$ такая, что $i \neq j, |a_i - a_j| \leqslant t$.
- Задача разрешимости не легче основной задачи.
- Сведённая задача решается сортировкой за $O(n \log n)$ и линейной проверкой разности соседей, т. е. $O(n \log n + n)$.
- Это сведение пока *не* доказывает нижнего предела поиска ближайшей пары.
- Если бы мы знали, что если в наихудшем случае близкую пару можно найти за $\Omega(n \log n)$, то это было бы доказательством, что сортировку нельзя провести быстрее.
- Если бы такая сортировка существовала бы, нахождение ближайшей пары было бы быстрее.

# Примеры сводимых задач

**Задача 2.** [Задача о вершинном покрытии] Имеется граф $G = (V, E)$ и целое число $k \leqslant |V|$. Существует ли такое $S \in V$, в котором не больше $k$ вершин, для которого каждое ребро $e \in E$ содержит по крайней мере одну вершину в множестве $S$?

**Задача 3.** [Задача о независимом множестве] Имеется граф $G = (V, E)$ и целое число $k \leqslant |V|$. Существует ли в графе $G$ независимое множество из $k$ вершин? Множество $S$ графа $G$ является независимым, если ни одна из вершин множества не соединена ребром с другой.
Задача о вершинном покрытии сводится к задаче о независимом множестве переходом $k'_{cover} = |V| - k_{set}$

# Примеры сводимых задач

**Задача 4.** [Задача календарного планирования] Имеется наборов набор линейных интервалов и целое число $k$. Можно ли из $I$ выбрать подмножество из не менее $k$ непересекающихся интервалов?

Задача о независимом множестве сводится к задаче календарного планирования. Для каждого из рёбер графа создаётся интервал. Каждая пара вершин с общим ребром определяет пару задач с общим интервалом. Наибольшие подмножества одинаковые. Если есть быстрый алгоритм для решения задачи календарного планирования — есть и для задачи о независимом множестве.

# Примеры сводимых задач

**Задача 5.** [Задача о клике] Имеется граф $G = (V, E)$ и целое число $k \leqslant |V|$. Содержит ли граф клику размером $k$?

Задача о независимом множестве сводится к задаче о клике. Создаём *дополнение* графа $G$ задачи о независимом множестве: если в $G$ вершины $u$ и $v$ были соединены ребром, то в $G'$ они не должны быть соединены и наоборот. Решаем задачу о клике для графа $G'$.
Если есть быстрый алгоритм для решения задачи о клике — есть и для независимом множестве.

# Приближённое решение NP-сложных задач.

# Задача о вершинном покрытии

- Пока множество рёбер $E$ не пусто
  - Выбираем произвольное ребро $e(u, v) \in E$
  - Добавляем $u$ и $v$ к множеству вершинного покрытия
  - Удаляем из $E$ все рёбра, инцидентные $u$ и $v$.
- Алгоритм корректный.
- Алгоритм жадный.
- Алгоритм быстрый.
- В худшем случае результат в 2 раза хуже верного.

# Задача коммивояжёра

**Задача 6.** Найти в графе $G$ в евклидовом пространстве гамильтонов путь наименьшей стоимости.

- Строим минимальное остовное дерево.
- Вес этого дерева $W$ — точная нижняя граница решения задачи.
- $DFS$-обход дерева даст как максимум удвоенную стоимость $W$, но вершины будут повторяться, $W' \leqslant 2W$.
- Для удаления лишних вершин на каждом шаге пойдём к каждой ближайшей непосещённой вершине.
- Правило треугольника обеспечивает сокращение пути относительно $W'$.
- Решение не хуже $2W$.

# Задача о соединении точек на плоскости

**Задача 7.** Имеется $2n$ точек на плоскости. Требуется разбить точки на ровно $n$ пар так, чтобы сумма длин отрезков, соединяющих точки, была минимальной.

# Решение 1. Полный перебор.

1. Нумеруем точки в произвольном порядке.
2. Выбираем первую точку множества.
3. Для всех точек множества, с ней не совпадающих выбираем ту, что выдаст наименьшее значение:

3.1 проводим отрезок, обновляем сумму;

3.2 удаляем обе точки из множества;

3.3 рекурсивно запускаем решение подзадачи на меньшем множестве.

# Решение 1. Полный перебор: сложность

- На первом шаге для первой точки мы переберём $2n - 1$ подзадач.
- Размер каждой подзадачи равен $2n - 2$.
- Общая сложность

$$T = O((2n - 1) \cdot (2n - 3) \cdot \dots \cdot 1) = O(n!!)$$

# Решение 2. Перебор с отсечением

- Давайте найдём какое-то решение.
- Пусть его результатом будет $R$.
- Так как все отрезки положительной ненулевой длины, при возврате из рекурсиии (спуску по дереву решений) результат только увеличивается.
- Искать решение, если при спуске по дереву уже достигнут результат $R' \geqslant R$ не имеет смысла.
- Текущая ветка *отсекается (prune)*.
- Количество узлов в переборе сокращается.

# Решение 3. Перебор с отсечением: раннее отсечение

- При отсечении выгодно как можно быстрее получить минимальный результат.
- Эвристика: начинать перебор с самой близкой точки.

| Точек | 14 | 16 | 18 | 20 | 20 | 24 |
|---|---|---|---|---|---|---|
| Full | 0.067 | 0.880 | 14.818 | 281.416 | - | - |
| Pruning | 0.004 | 0.008 | 0.045 | 0.792 | 0.368 | 11.042 |
| Early pruning | 0.004 | 0.004 | 0.017 | 0.466 | 0.159 | 4.085 |

# Решение 4. Вероятностные методы

- Сложность решения перебором с отсечением или без него факториальная и решение точное.
- Можно сложность уменьшить до экспоненциальной, рассматривая не всех кандидатов, а только несколько первых.
- Это тоже отсечение ветвей, но под него могут попасть и ветки с решением.

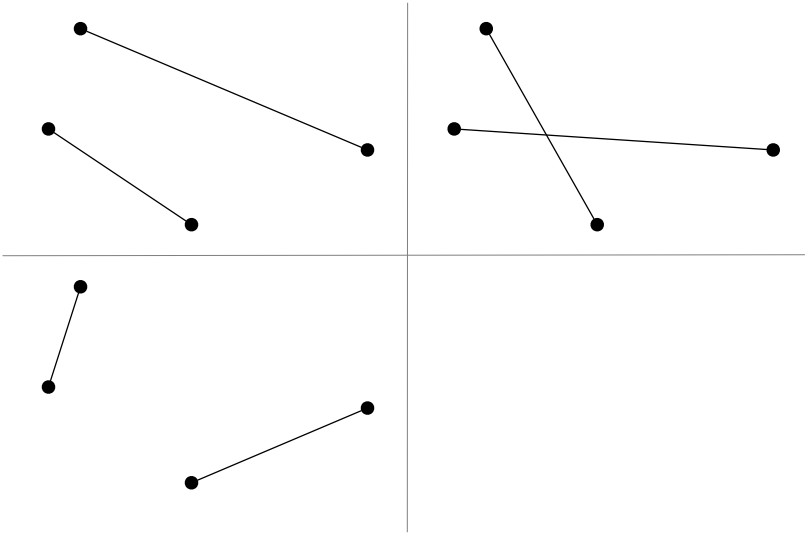
Перебираем не более трёх ближайших точек.

| Точек | 24 | 24 | 26 | 26 | 28 | 30 |
|---|---|---|---|---|---|---|
| Early pruning,t | 6.088 | 5.169 | 53.761 | 18.943 | 19.944 | 194.430 |
| result | 16.779 | 17.227 | 18.189 | 17.498 | 15.479 | 18.062 |
| Cut 3,t | 0.025 | 0.022 | 0.045 | 0.072 | 0.040 | 0.268 |
| result | 16.779 | 17.227 | 18.892 | 17.677 | 15.479 | 19.298 |

# Решение 5. Вероятностные методы

- Мы использовали вероятность того, что идеальное решение находится в неотсекаемой части.
- Давайте пойдём дальше и увеличим случайность.
- Пусть уже есть построенное решение.
- Возьмём две случайных пары и попытаемся создать из них на этих же четырёх точках две других.
- Если существует другая комбинация, дающая лучшее решение, переформируем пары.

# Переформировывание пар



| Точек | 30 | 32 | 34 | 36 | 38 | 40 |
|---|---|---|---|---|---|---|
| Cut 3,t | 0.185 | 0.971 | 1.379 | 5.029 | 8.328 | 10.227 |
| result | 18.055 | 24.964 | 21.074 | 19.129 | 23.023 | 22.877 |
| Evol,t | 0.006 | 0.006 | 0.006 | 0.006 | 0.006 | 0.006 |
| result | 18.350 | 25.390 | 21.074 | 19.129 | 23.897 | 21.839 |

# Решение обратных и многопараметрических задач

Пусть математическая модель явления характеризуется вектором $\overrightarrow{m}$, принадлежащим некоторому пространству моделирования $M$. Пусть $\overrightarrow{d}$ означает некоторые атрибуты явления, $\overrightarrow{d} \in D$, где $D$ — пространство данных. Имеется некоторый оператор $A$, для которого

$$\overrightarrow{d} = A(\overrightarrow{m}),$$

который связывает параметры модели с данными.
Математическая сложность решения обратных задач заключается в том, что обратный оператор $A^{-1}$ может быть либо разрывным, либо вообще не существовать. Одним из способов решения обратной задачи является переход от поиска обратного оператора к некоторой задаче минимизации:

$$\overrightarrow{d} = A(\overrightarrow{m}) \Leftrightarrow \min_{\overrightarrow{m} \in M} \|d - A(\overrightarrow{m})\|,$$

где в качестве нормы может быть взята например норма $L_1, L_2$ или $L_\infty$.
Таким образом решение обратной задачи сводится к многократному решению прямой задачи.

# Методы решения экстремальных задач

Основные методы поиска экстремума делятся на направленные, ненаправленные и переборные.

- **Градиентные методы** Пробная точка перемещается в допустимой области значений аргументов в направлении, связанном с антиградиентом.
- Градиентные методы хорошо сходятся, если целевая функция гладкая, дифференцируема на всем пространстве поиска и имеет один экстремум на пространстве поиска. Проблемы при решении задач нахождения глобального экстремума многопараметрических сложно-вычисляемых функций:
  - нахождение градиента функции в точке требует вычисления по меньшей мере $2N$ значений функции в точке, где $N$-размерность пространства параметров;
  - градиентные методы имеют тенденцию сходиться к локальным минимумам целевых функций;
  - даже при наличии гладких функций градиентные методы при определённых условиях могут потребовать большого количества итераций для достижения условия сходимости;
  - градиентные методы чувствительны к зашумлённости значений целевой функции

# Эволюционные методы

- Эволюционные методы используют элементы случайного поиска в пространстве задачи с привнесением элементов детерминированности.
- Суть эволюционных методов — отбор наилучших объектов.
- Время поиска экстремума по сравнению с чисто вероятностными методами может быть уменьшено на несколько порядков.
- Детерминированность заключается в моделировании природных процессов отбора, размножения и наследования, происходящих по определенным правилам.
- В качестве случайного элемента выступает, например, мутация.

# Метод дифференциальной эволюции

- Генерация начальной популяции.
- Воспроизводство потомков.
- Мутация.
- Отбор.

# Генерация начальной популяции

- Выбирается $N$ произвольных векторов $X_1, X_2, \ldots, X_N$ из пространства $R^N$, в котором находится целевая функция, требующая минимизации, называемые *начальной популяцией*.
- Их выбор обычно связан с постановкой задачи, если известен априорный вид функции, или её некоторые свойства, либо вектора имеют равномерное или нормальное распределение.

# Воспроизводство потомков

- Для любого вектора $T$ из старой популяции берётся три произвольных вектора из этого же поколения: $X_i, X_j, X_k$, и по ним вычисляется мутантный вектор $V$ по формуле: $V = X_i + F(X_j - X_k)$.
- Параметр $F$ — один из параметров метода — это так называемая *сила мутации*.
- В методе дифференциальной эволюции используется внутренний источник шума — разность между случайными представителями поколения.

# Мутация

- На следующем шаге метода с вектором $V$ производятся мутации:
- пусть $p$ — один из параметров метода, *вероятность мутации*, с которой потомок $W$ наследует какой-либо из признаков родителя $T$.
- $W_i = T_i$ с вероятностью $p$ и $W_i = V_i$ с вероятностью $1 - p$ для каждого признака $i = 1, \ldots, n$.
- Таким образом, каждый вектор из популяции претерпевает путь $T \to V \to W$.

# Отбор

- После вычисления вектора-потомка $W$ сравниваются значения целевой функции для него и для его родителя $T$.
- В новой популяции остаётся тот, на котором целевая функция принимает меньшее значение.
- Таким образом, размер популяции при этом не меняется, и каждый раз «выживает сильнейший», более приспособленный вектор.
- Разумеется, существует вероятность того, что популяцию покинет вектор, который через несколько поколений привёл бы к лучшему результату. Вероятность этого уменьшает правильный выбор размера популяции.

# Особенности метода

- Позволяет динамически моделировать особенности рельефа оптимизируемой функции. Источник шума — разность между случайно выбранными векторами текущей популяции.
- Плотность распределения популяции выше вблизи локальных минимумов функции.
- Появление точек популяции вблизи глобального минимума приводит к увеличению вероятности миграции точек из зон локального минимума.
- Малый размер популяции — успевается создать большое количество поколений, но вероятность схождения к *локальному* экстремуму повышается.
- Слишком большой размер популяции может привести к тому, что число поколений станет недостаточным для нахождения глобального экстремума.
- Большая сила мутации $F$ — увеличивается стохастическая составляющая алгоритма. Локальная структура целевой функции при этом практически не используется.
- Небольшая сила мутации — алгоритм приближается к градиентным методам, так как за счет подстройки облака точек к структуре функции он фактически строит приближение градиента.