Введение в экспоненциальные алгоритмы

Лекция N 1 курса "Современные задачи теоретической информатики"

Юрий Лифшиц
yura@logic.pdmi.ras.ru

ИТМО

Осень'2005

— Г-голубчики, — сказал Федор Симеонович озадаченно... — Это же проблема Бен Б-бецалеля. К-калиостро же доказал, что она н-не имеет р-решения.

— Мы сами знаем, что она не имеет решения, — сказал Хунта, немедленно ощетиниваясь. — Мы хотим знать, как ее решать.

— К-как-то ты странно рассуждаешь, К-кристо... К-как же искать решение, к-когда его нет? Б-бессмыслица какая-то...

— Извини, Теодор, но это ты очень странно рассуждаешь. Бессмыслица — искать решение, если оно и так есть. Речь идет о том, как поступать с задачей, которая решения не имеет. Это глубоко принципиальный вопрос...

А. и Б. Стругацкие, Понедельник начинается в субботу

План лекции

1. Общая идеология
2. Рекурсия
   - 1.84^n для 3-SAT
   - 1.89^n для 3-раскрашиваемости
3. Локальный поиск
4. Вероятностные алгоритмы
   - Сведение к полиномиальной задаче
   - Изменение порядка шагов
   - Вероятностный локальный поиск
5. Задачи

Время работы алгоритма

Полиномиальные алгоритмы

n^2 $n \log n$ n^3 n^{1000}

Время работы алгоритма

Полиномиальные алгоритмы

n^2 $n \log n$ n^3 n^{1000}

Псевдополиномиальные алгоритмы

$n^{\log n}$ $n^{\log^2 n}$ $c^{\log^7 n}$

Время работы алгоритма

Полиномиальные алгоритмы

n^2 $n \log n$ n^3 n^{1000}

Псевдополиномиальные алгоритмы

$n^{\log n}$ $n^{\log^2 n}$ $c^{\log^7 n}$

Субэкспоненциальные алгоритмы

$2^{\sqrt{n}}$ $5^{(n^{0.98})}$

Время работы алгоритма

Полиномиальные алгоритмы

n^2 $n \log n$ n^3 n^{1000}

Псевдополиномиальные алгоритмы

$n^{\log n}$ $n^{\log^2 n}$ $c^{\log^7 n}$

Субэкспоненциальные алгоритмы

$2^{\sqrt{n}}$ $5^{(n^{0.98})}$

Экспоненциальные алгоритмы

2^n 8^n $n!$ n^n

Время работы алгоритма

Полиномиальные алгоритмы

n^2 $n \log n$ n^3 n^{1000}

Псевдополиномиальные алгоритмы

$n^{\log n}$ $n^{\log^2 n}$ $c^{\log^7 n}$

Субэкспоненциальные алгоритмы

$2^{\sqrt{n}}$ $5^{(n^{0.98})}$

Экспоненциальные алгоритмы

2^n 8^n $n!$ n^n

Сравните 1.1^n и $10n^3$ для n=100

“Hardware” vs. “Brainware”

Базовый алгоритм:

Время работы 2^n, на практике работает для $n \leq n_0$

“Hardware” vs. “Brainware”

Базовый алгоритм:

Время работы 2^n, на практике работает для $n \leq n_0$

Аппаратный подход (закон Мура)

За полтора года компьютеры удвоили производительность

“Hardware” vs. “Brainware”

Базовый алгоритм:

Время работы 2^n, на практике работает для $n \leq n_0$

Аппаратный подход (закон Мура)

За полтора года компьютеры удвоили производительность

$$n_0 \longrightarrow n_0 + 1$$

“Hardware” vs. “Brainware”

Базовый алгоритм:

Время работы 2^n, на практике работает для $n \leq n_0$

Аппаратный подход (закон Мура)

За полтора года компьютеры удвоили производительность

$$n_0 \longrightarrow n_0 + 1$$

Мозговой подход

Удалось придумать алгоритм работающий за 1.41^n

“Hardware” vs. “Brainware”

Базовый алгоритм:

Время работы 2^n, на практике работает для $n \leq n_0$

Аппаратный подход (закон Мура)

За полтора года компьютеры удвоили производительность

$$n_0 \longrightarrow n_0 + 1$$

Мозговой подход

Удалось придумать алгоритм работающий за 1.41^n

$$n_0 \longrightarrow 2n_0$$

“Hardware” vs. “Brainware”

Базовый алгоритм:

Время работы 2^n, на практике работает для $n \leq n_0$

Аппаратный подход (закон Мура)

За полтора года компьютеры удвоили производительность

$$n_0 \longrightarrow n_0 + 1$$

Мозговой подход

Удалось придумать алгоритм работающий за 1.41^n

$$n_0 \longrightarrow 2n_0$$

Кто такой Гордон Мур?

Класс NP

Класс NP

Неформально: задачи, которые могут быть решены перебором $2^{poly(n)}$ вариантов

Принадлежность классу **NP** — это положительная характеристика!

Пример: разложение числа на множители

Класс NP

Класс NP

Неформально: задачи, которые могут быть решены перебором $2^{poly(n)}$ вариантов

Принадлежность классу **NP** — это положительная характеристика!

Пример: разложение числа на множители

NP-полные задачи

Неформально: наиболее трудные задачи внутри класса **NP**

Примеры: Коммивояжер, Гамильтонов цикл, Сумма размеров

Класс NP

Класс NP

Неформально: задачи, которые могут быть решены перебором $2^{poly(n)}$ вариантов

Принадлежность классу **NP** — это положительная характеристика!

Пример: разложение числа на множители

NP-полные задачи

Неформально: наиболее трудные задачи внутри класса **NP**

Примеры: Коммивояжер, Гамильтонов цикл, Сумма размеров

NP-трудные задачи

Неформально: задачи к которым можно полиномиально свести любою задачу из класса **NP**

Не обязательно лежат в **NP**!

Пример: задача об остановке

Новые идеи

Методы для решения NP-полных задач

- Рекурсия

Новые идеи

Методы для решения NP-полных задач

- Рекурсия
- Хитрый перебор

Новые идеи

Методы для решения NP-полных задач

- Рекурсия
- Хитрый перебор
- Локальный поиск

Новые идеи

Методы для решения NP-полных задач

- Рекурсия
- Хитрый перебор
- Локальный поиск
- Случайный порядок действий

Новые идеи

Методы для решения NP-полных задач

- Рекурсия
- Хитрый перебор
- Локальный поиск
- Случайный порядок действий
- Вероятностное сведение к простой задаче

Новые идеи

Методы для решения NP-полных задач

- Рекурсия
- Хитрый перебор
- Локальный поиск
- Случайный порядок действий
- Вероятностное сведение к простой задаче
- Случайное блуждание

Новые идеи

Методы для решения NP-полных задач

- Рекурсия
- Хитрый перебор
- Локальный поиск
- Случайный порядок действий
- Вероятностное сведение к простой задаче
- Случайное блуждание
- На следующей лекции: разделяй и властвуй

SAT и 3-SAT

Задача ВЫПОЛНИМОСТЬ (SAT)

Дано: булева формула в КНФ

Определить: существует ли выполняющая подстановка?

SAT и 3-SAT

Задача ВЫПОЛНИМОСТЬ (SAT)

Дано: булева формула в КНФ

Определить: существует ли выполняющая подстановка?

Выполнима ли формула: $(\neg a \vee \neg b)\&(a \vee b \vee c)\&(a \vee \neg b)$?

SAT и 3-SAT

Задача ВЫПОЛНИМОСТЬ (SAT)

Дано: булева формула в КНФ

Определить: существует ли выполняющая подстановка?

Выполнима ли формула: $(\neg a \vee \neg b)\&(a \vee b \vee c)\&(a \vee \neg b)$?

Задача 3-ВЫПОЛНИМОСТЬ (3-SAT)

Каждая скобка содержит не более трех переменных

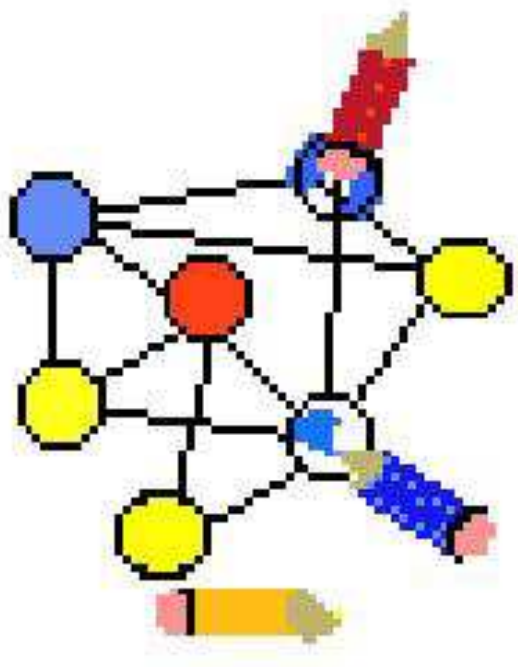
Факт: 3-SAT является **NP**-полной

3-раскрашиваемость

Задача 3-раскрашиваемость

Дано: неориентированный граф

Определить: существует ли правильная раскраска вершин в три цвета?



План лекции

1. Общая идеология
2. **Рекурсия**
   - 1.84^n для 3-SAT
   - 1.89^n для 3-раскрашиваемости
3. Локальный поиск
4. Вероятностные алгоритмы
   - Сведение к полиномиальной задаче
   - Изменение порядка шагов
   - Вероятностный локальный поиск
5. Задачи

Рекурсия для 3-SAT

Идея:

Сводить решение задачи для n переменных к меньшему количеству переменных

Рекурсия для 3-SAT

Идея:

Сводить решение задачи для n переменных к меньшему количеству переменных

Алгоритм Мониена и Шпекенмайера [1985]

1. Фиксируем скобку (пусть $(x \vee y \vee z)$)

Рекурсия для 3-SAT

Идея:

Сводить решение задачи для n переменных к меньшему количеству переменных

Алгоритм Мониена и Шпекенмайера [1985]

1. Фиксируем скобку (пусть $(x \vee y \vee z)$)
2. Решаем задачу для $x = 1$

Рекурсия для 3-SAT

Идея:

Сводить решение задачи для n переменных к меньшему количеству переменных

Алгоритм Мониена и Шпекенмайера [1985]

1. Фиксируем скобку (пусть $(x \vee y \vee z)$)
2. Решаем задачу для $x = 1$
3. Если ответ "невыполнима", решаем задачу для $x = 0, y = 1$

Рекурсия для 3-SAT

Идея:

Сводить решение задачи для n переменных к меньшему количеству переменных

Алгоритм Мониена и Шпекенмайера [1985]

1. Фиксируем скобку (пусть $(x \vee y \vee z)$)
2. Решаем задачу для $x = 1$
3. Если ответ "невыполнима", решаем задачу для $x = 0, y = 1$
4. Если ответ "невыполнима", решаем задачу для $x = 0, y = 0, z = 1$

Рекурсия для 3-SAT

Идея:

Сводить решение задачи для n переменных к меньшему количеству переменных

Алгоритм Мониена и Шпекенмайера [1985]

1. Фиксируем скобку (пусть $(x \vee y \vee z)$)
2. Решаем задачу для $x = 1$
3. Если ответ "невыполнима", решаем задачу для $x = 0, y = 1$
4. Если ответ "невыполнима", решаем задачу для $x = 0, y = 0, z = 1$

Рекурсия для 3-SAT

Идея:

Сводить решение задачи для n переменных к меньшему количеству переменных

Алгоритм Мониена и Шпекенмайера [1985]

1. Фиксируем скобку (пусть $(x \vee y \vee z)$)
2. Решаем задачу для $x = 1$
3. Если ответ "невыполнима", решаем задачу для $x = 0, y = 1$
4. Если ответ "невыполнима", решаем задачу для $x = 0, y = 0, z = 1$

Рекурентное уравнение:

$T(n) \leq T(n-1) + T(n-2) + T(n-3)$

Факт: $T(n) = O(1.84^n)$

Рекурсия для раскраски

Тупая идея: полный перебор — $O(3^n)$

Рекурсия для раскраски

Тупая идея: полный перебор — $O(3^n)$

Простой рекурсивный алгоритм:

Фиксируем цвет первой вершины

Перебираем не более чем ДВА варианта раскраски какого-нибудь соседа для уже раскрашенных

Сложность - $O(2^n)$

Рекурсия для раскраски

Тупая идея: полный перебор — $O(3^n)$

Простой рекурсивный алгоритм:

Фиксируем цвет первой вершины

Перебираем не более чем ДВА варианта раскраски какого-нибудь соседа для уже раскрашенных

Сложность - $O(2^n)$

Можно сделать хитрее:

Факт: 2-раскрашиваемость полиномиально разрешима

В один из цветов покрашено не более $n/3$ вершин

Переберем все варианты для множества синих вершин:

$$\sum_{i=0}^{n/3} C_n^i \leq 1.89^n$$

Рекурсия для раскраски

Тупая идея: полный перебор — $O(3^n)$

Простой рекурсивный алгоритм:

Фиксируем цвет первой вершины

Перебираем не более чем ДВА варианта раскраски какого-нибудь соседа для уже раскрашенных

Сложность - $O(2^n)$

Можно сделать хитрее:

Факт: 2-раскрашиваемость полиномиально разрешима

В один из цветов покрашено не более $n/3$ вершин

Переберем все варианты для множества синих вершин:

$$\sum_{i=0}^{n/3} C_n^i \leq 1.89^n$$

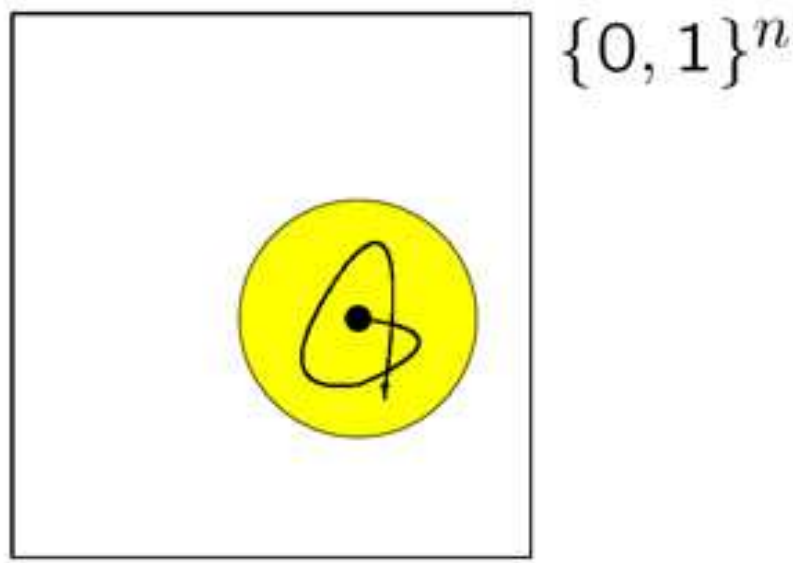
Полиномиальный алгоритм для 2-раскрашиваемости?

План лекции

Локальный поиск

Новая идея:

Взять подстановку и начать ее “улучшать”



Локальный поиск II

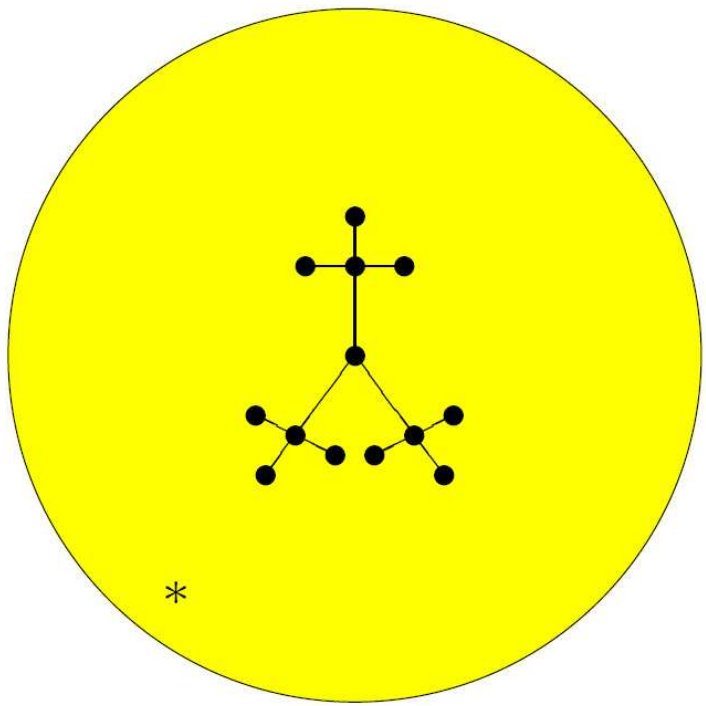
Алгоритм улучшения

Рассматриваем невыполненную скобку

Пытаемся (рекурсивно) изменить одну из переменных

Для глубины d — сложность $O(3^d)$

Если наш набор отличался от правильного значениями d переменных — мы решим задачу!



Локальный поиск II

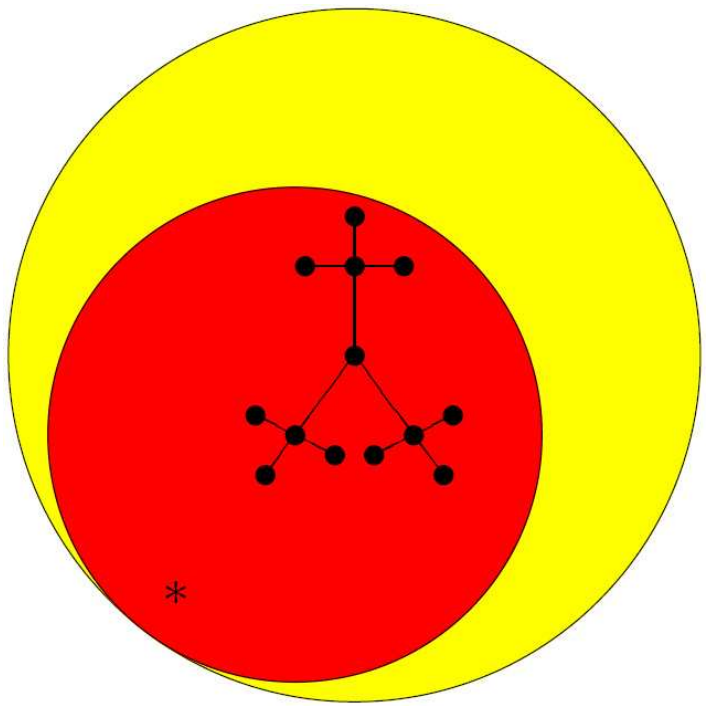
Алгоритм улучшения

Рассматриваем невыполненную скобку

Пытаемся (рекурсивно) изменить одну из переменных

Для глубины d — сложность $O(3^d)$

Если наш набор отличался от правильного значениями d переменных — мы решим задачу!



Локальный поиск III

Начальный набор

Возьмем два: 0^n и 1^n

Для одного из них $d \leq n/2!$

Сложность алгоритма $3^{n/2} = 1.74^n$

Локальный поиск III

Начальный набор

Возьмем два: 0^n и 1^n

Для одного из них $d \leq n/2!$

Сложность алгоритма $3^{n/2} = 1.74^n$

Дальнейшее ускорение алгоритма

Покрывающие коды

План лекции

1. Общая идеология
2. Рекурсия
   - 1.84^n для 3-SAT
   - 1.89^n для 3-раскрашиваемости
3. Локальный поиск
4. **Вероятностные алгоритмы**
   - Сведение к полиномиальной задаче
   - Изменение порядка шагов
   - Вероятностный локальный поиск
5. Задачи

Вероятностные алгоритмы

Алгоритмы с двусторонней ошибкой:

Если ответ “Да”, алгоритм говорит “Да” с вероятностью хотя бы 2/3

Если ответ “Нет”, алгоритм говорит “Нет” с вероятностью хотя бы 2/3

Вероятностные алгоритмы

Алгоритмы с двусторонней ошибкой:

Если ответ “Да”, алгоритм говорит “Да” с вероятностью хотя бы 2/3

Если ответ “Нет”, алгоритм говорит “Нет” с вероятностью хотя бы 2/3

Алгоритмы с односторонней ошибкой:

Если ответ “Да”, алгоритм говорит “Да”

Если ответ “Нет”, алгоритм говорит “Нет” с вероятностью хотя бы 2/3

Вероятностные алгоритмы

Алгоритмы с двусторонней ошибкой:

Если ответ “Да”, алгоритм говорит “Да” с вероятностью хотя бы 2/3

Если ответ “Нет”, алгоритм говорит “Нет” с вероятностью хотя бы 2/3

Алгоритмы с односторонней ошибкой:

Если ответ “Да”, алгоритм говорит “Да”

Если ответ “Нет”, алгоритм говорит “Нет” с вероятностью хотя бы 2/3

Алгоритмы с нулевой ошибкой:

С вероятностью хотя бы 2/3 — правильный ответ

В остальных случаях алгоритм говорит “не знаю”

Вероятностные алгоритмы II

Общая схема для задач поиска:

Найти алгоритм работающий за время t

Оценить вероятность успеха p

Повторять исходный алгоритм $c \cdot p^{-1}$ раз

Итоговая вероятность станет константой:

$$(1-p)^{c/p} \leq e^{-c}$$

Вероятностные алгоритмы II

Общая схема для задач поиска:

Найти алгоритм работающий за время t

Оценить вероятность успеха p

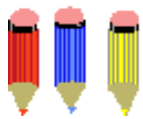
Повторять исходный алгоритм $c \cdot p^{-1}$ раз

Итоговая вероятность станет константой:

$$(1-p)^{c/p} \leq e^{-c}$$

К какому типу относится эта схема?

Рандомизация для раскраски



Общая схема:

С вероятностью 2/3 угадать НЕ цвет

Факт: раскрашиваемость узнается за $poly(n)$

Вероятность успеха $(2/3)^n$

Сложность итогового алгоритма $O(1.5^n)$!

Изменение порядка шагов

Paturi и другие [1997]

Общая схема:

Выбрать случайный порядок переменных

Делать присвоения согласно этому порядку

Если находится скобка с одной переменной — использовать ее.

Анализ: в среднем глубина рекурсии составляет $2n/3$!

Итоговая сложность $O(2^{2n/3}) = O(1.58^n)$

Локальный поиск

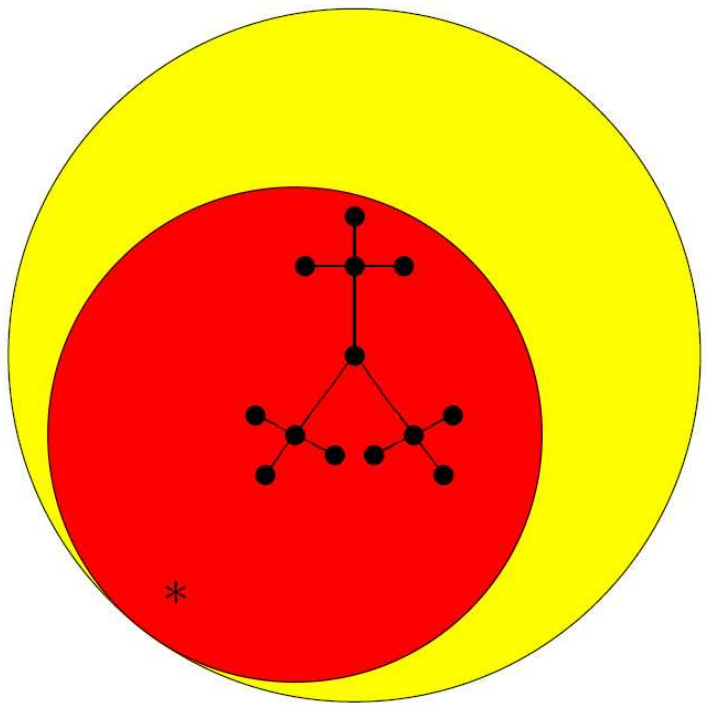
Вспоминаем детерминированный поиск:

Рассматриваем невыполненную скобку

Пытаемся (рекурсивно) изменить одну из переменных

Для глубины d — сложность $O(3^d)$

Если наш набор отличался от правильного значениями d переменных — мы решим задачу!

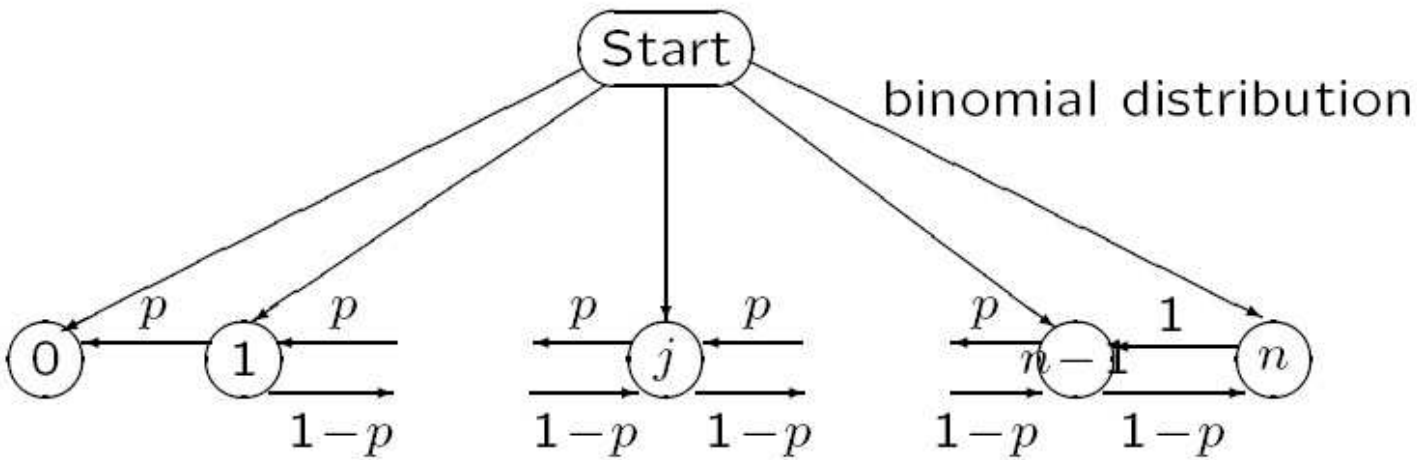


Вероятноятностный метод

Новая идея:

Делать поиск вероятностным

Удлиннить поиск до $3d$



Вероятность становится *почти* $(1/2)^d$!

Оценка времени работы

Исходный набор выберем случайным образом

Математическое ожидание работы алгоритма:

$$E[(1/2)^{X_1+\cdots+X_n}] = \prod_{i=1}^{n} E[(1/2)^{X_i}] = (3/4)^n$$

Оценка времени работы

Исходный набор выберем случайным образом

Математическое ожидание работы алгоритма:

$$E[(1/2)^{X_1+\cdots+X_n}] = \prod_{i=1}^{n} E[(1/2)^{X_i}] = (3/4)^n$$

Время работы алгоритма — $(4/3)^n = 1.33^n$!

План лекции

1. Общая идеология
2. Рекурсия
   - 1.84^n для 3-SAT
   - 1.89^n для 3-раскрашиваемости
3. Локальный поиск
4. Вероятностные алгоритмы
   - Сведение к полиномиальной задаче
   - Изменение порядка шагов
   - Вероятностный локальный поиск
5. **Задачи**

Задачи для размышления

Найдите математическое ожидание “бесплатных присвоений” в алгортме Paturi

Задачи для размышления

Найдите математическое ожидание “бесплатных присвоений” в алгортме Paturi

Докажите, что предельная вероятность для цепи $(1/3, 2/3)$ равна 2^{-d}

Задачи для размышления

Найдите математическое ожидание “бесплатных присвоений” в алгортме Paturi

Докажите, что предельная вероятность для цепи $(1/3, 2/3)$ равна 2^{-d}

Постройте полиномиальный алгоритм для 2-раскрашиваемости

Последний слайд

Если не запомните ничего другого:

- Новые методы для экспоненциальных алгоритмов

Последний слайд

Если не запомните ничего другого:

- Новые методы для экспоненциальных алгоритмов
- Особенно много вероятностных алгоритмов

Последний слайд

Если не запомните ничего другого:

- Новые методы для экспоненциальных алгоритмов
- Особенно много вероятностных алгоритмов
- Новые принципы вероятностных алгоритмов

Последний слайд

Если не запомните ничего другого:

- Новые методы для экспоненциальных алгоритмов
- Особенно много вероятностных алгоритмов
- Новые принципы вероятностных алгоритмов

Последний слайд

Если не запомните ничего другого:

- Новые методы для экспоненциальных алгоритмов
- Особенно много вероятностных алгоритмов
- Новые принципы вероятностных алгоритмов

Вопросы?