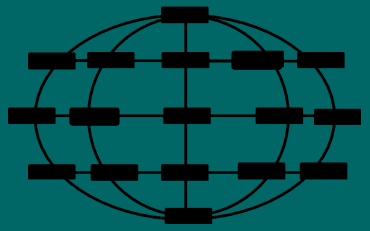
Методы построения и анализа алгоритмов



Лекция 5

Малышкин Виктор Эммануилович

Кафедра Параллельных Вычислительных Технологий
Новосибирский государственный технический университет

E_mail: malysh@ssd.sscc.ru
Телефон: 3308 994

Новосибирск

How to design algorithms

The questions that should be answered by a designer.

- **I. Понятна ли проблема?**
- (а) что точно входит во входные данные?
- (б) что есть желаемый результат?
- (в) Можно ли сконструировать небольшой пример входных данных такой, чтобы вручную решить проблему? Что происходит в ходе решения?
- (г) всегда ли нужно оптимальное решение? Может ли пригодиться субоптимальное решение?

(д) насколько велика типичная проблема? Как велики обрабатываемые данные? 10 или 100 или 10000 единиц данных? Ограничена проблема или неограниченна?

(е) насколько важно время решения проблемы? Одна секунда? Одна минута? Час или день? Год? Другие качества?

(ж) Сколько времени и усилий можно потратить на решение проблемы? Должно ли ограничиться простым алгоритмом, который м.б. запрограммировать за день? Неделю? Или же есть время поэкспериментировать с 2-3 алгоритмами (подходами) и выбрать из них лучший?

(з) Какой тип проблемы решается? Графическая? Геометрическая? Численная? Теоретико-множественная? Какая формулировка проще?

II. Можно ли найти простой алгоритм или эвристику для решения проблемы?

- (a) годится ли для решения полный переборный алгоритм или же нужна будет некоторая оптимизирующая организация данных?

(i) есть ли уверенность в корректности результата?

(ii) как замеряется качество решения?

(iii) работает этот простой медленный алгоритм за полиномиальное или экспоненциальное время?

(iv) есть ли уверенность, что проблема достаточно хорошо определена, чтобы решение было корректным?

- (б)можно ли решить проблему, многократно применяя некоторый простой прием, например, выбирать наибольший элемент первым? Наименьший элемент первым? Произвольный элемент?

(i) если так, то на каких входных данных эвристика работает хорошо? Это те самые данные, которые будут встречаться в требуемом приложении?

(ii) на каких данных эвристика работает плохо? Или же всегда работает хорошо?

(iii) Насколько проста эвристика в реализации? Насколько быстро вырабатывает решение?

III. Можно ли найти алгоритм решения проблемы?

- (а) Что в мире известно о проблеме? Есть ли доступная реализация?
- (б) Просмотрены ли все доступные материалы по проблеме?
- (в) Нет ли в Интернете каких-то подходящих сайтов?

IV.Есть ли особый случай проблемы, который я знаю как решить?

- (а) можно ли решить проблему, проигнорировав некоторые из входных параметров?
- (б) будет ли проблема решаться легче, если присвоить некоторым входным параметрам конкретные, м.б. тривиальные, значения?
- (в) можно ли так упростить проблему чтобы ее можно было решить эффективно?
- (г) почему решение для упрощенного случая не может быть обобщено на более широкий класс входов?
- (д) Не является ли проблема частным случаем более общей проблемы?

V. Какой из известных способов конструирования алгоритма наиболее подходит к решаемой проблеме?

(а) Существует ли множество элементов, которое может быть упорядочено по какому-то ключу? Упрощает ли такое упорядочивание решение проблемы?

(б) Существует ли способ разделить проблему на две меньшие? На большие и меньшие? Левые и правые? Можно ли применить стратегию разделяй-и-властвуй (merge) (divide –and- conquer)?

(в) Существует ли естественный порядок на входных и/или выходных множествах (строки, перестановки, листья в деревьях и т.п.) такой, что может быть применено динамическое программирование?

Динамическое программирование

Динамическое программирование — способ решения сложных задач путём разбиения их на более простые подзадачи, сложность которых меньше исходной

Чтобы решить поставленную задачу, требуется решить подзадачи, после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

- Метод *динамического программирования сверху* — это простое запоминание результатов решения тех подзадач, которые могут повторно встретиться в дальнейшем.
- *Динамическое программирование снизу* включает в себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач.
- Числа Фибоначчи $F_3=F_1+F_2$ и $F_4=F_2+F_3$

В общем случае задача решается в три шага

- Разбиение задачи на подзадачи меньшего размера.
- Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм
- Использование полученного решения подзадач для конструирования решения исходной задачи.

Для решения задач используются два подхода:

- нисходящее динамическое программирование: задача разбивается на подзадачи меньшего размера, они решаются и затем комбинируются для решения исходной задачи. Используется запоминание для решений часто встречающихся подзадач.
- восходящее динамическое программирование: все подзадачи, которые впоследствии понадобятся для решения исходной задачи просчитываются заранее и затем используются для построения решения исходной задачи. Этот способ лучше нисходящего программирования в смысле размера необходимых памяти и количества вызова функций, но иногда бывает нелегко заранее выяснить, решение каких подзадач нам потребуется в дальнейшем.

- (г) Можно ли использовать произвольный выбор последующего объекта?
- (д) Похожа ли проблема на одну из экспоненциальной сложности проблем? Может быть не существует ее эффективное решение? (Gary and Johnson)
- (е) Могут ли структуры данных быть использованы (очередь, словарь, хэш-таблицы, приоритеты и т.п.) для ускорения решения проблемы?

Рекомендуемые учебники

- Ахо, Альфред, В., Хопкрофт, Джон, Ульман, Джеффри, Д. *Структуры данных и алгоритмы*. : Пер. с англ. : Уч. пос. — М. : Издательский дом "Вильяме", 2000. — 384 с.
- Кормен Т., Лейзерсон Ч., Риверс Р., Штайн К. *Алгоритмы. Построение и анализ* – М.: «Вильямс», 2012
- В.Э.Малышкин, В.Д.Корнеев. *Параллельное программирование мультикомпьютеров*. – В серии «Учебники НГТУ», Новосибирск, изд-во НГТУ, 2011, 296 стр. (есть в библиотеке)

ВОПРОСЫ

1. Что мы называем алгоритмом? Почему?
2. Сколько существует алгоритмов и программ, вычисляющих вычислимую функцию?
3. Задача, ее модель, алгоритм решения
4. Задача управления движением на перекрестке и ее модель
5. Три подхода к решению комбинаторной задачи
6. Задача раскраски графа. Жадный алгоритм раскраски графа
7. Абстрактные типы данных. Что такое?

ВОПРОСЫ

8.Что такое вычислительная сложность алгоритма?

9. Время работы алгоритма. От чего зависит? Верхняя оценка сложности.

10. Общая схема решения *переборных* задач .Какие алгоритмы называются эвристическими?

11. Задача/проблемы построения расписания

12, Формулировки задачи построения расписания.

13. Способы сокращения перебора.

14. Стратегии построения субоптимпльных расписаний