算法分析与设计

上机实验选题及要求

1 实验目的

- 掌握利用算法进行问题求解的实验流程
- 强化对课堂所学算法原理的理解
- 提升利用算法原理进行编码实践的能力
- 了解算法研究的前沿动态，拓展算法改进的思路

2 实验内容及要求

实验采取自由分组的方式完成，每组至多 **3** 人，也可**独立完成**，编程语言推荐但不限于 **c/c++**。要求每组从下列主题中选择 **1 项**作为实验内容，完成实验设计、过程实现、结果分析、实验总结及实验报告撰写。

2.1 算法改进类

针对货箱装船问题、0/1 背包问题、子集和数问题、多段图最段路径问题、所有点对最短路径问题、矩阵乘法链问题、最大非交叉子网问题、最长公共子序列问题、带罚款额的作业调度问题、TSP 问题的、最近点对问题、数列逆序对问题 **其中之一**:

- 指出现有求解算法的局限，提出改进方案或新的算法解决方案
- 融合多种算法策略，针对问题特点，提出改进方案
- 放松约束条件的限制，提出改进方案
- 加强约束条件的限制，提出改进方案

要求：改进方案有针对性、思路清晰；需要对比实验结果。

2.2 算法验证类

- 利用贪心、分治、动态规划、回溯及分支限界中某一算法，选择附件 1 中至少两个问题的至少三个数据集进行实验验证.

要求：分析数据集分布特点对算法性能的影响，归纳算法解决不同问题的优缺点。

2.3 问题求解类

- 针对附件 1 中某一问题的**至少 *3* 个数据集**，利用利用贪心、分治、动态规划、回溯及分支限界中**至少三种**算法进行求解。

要求：分析不同算法解决同一问题的复杂度，归纳不同求解算法的优缺点。

2.4 前沿算法实现类

- 从附件 2 中选择 ***1* 篇文献**，翻译原文，编码实现其中的核心算法，并进行实验验证。

要求：必须以 PPT 的方式展示算法的基本原理、核心算法的实现细节。

3 实验报告要求

- 题目：60 字以内，如：“动态规划算法问题求解性能比较”
- 摘要：300 字左右，如：“本实验针对... 问题，利用... 算法，对... 数据集进行性能分析。实验结果表明...”
- 正文：5-8 页，包括实验目的、实验设计流程、代码实现、实验结果及复杂性分析
- 总结：200 字左右，包括实验过程总结和得出的主要结论。

4 实验文档递交

实验文档需按图1的方式组织，压缩（建议使用 Info-Zipurl系列进行压缩）后按学号-任务序号-成员数目.zip的方式命名 (如3021999999-1-3.zip)，若小组成员多于两人，学号为**组长学号**，每个小组只需递交 1 份，不迟于截止日期前递交到**智慧树**。

递交实验文档时务必注意以下几点：

① 根目录必须按学号-任务序号-成员数目命名，如3021999999-1-3。

② 根目录下必须包括四个子目录，分别为 src, data, experiments, reports。

③ 根目录下必须包含readme.md和member.txt两个文件，已给出相应的模版。

④ reports 目录下有三个文件分别为实验报告、讲解 ppt以及5-10 分钟讲解视频，不需改变文件名。

⑤ 程序文件建议以算法名称 + 问题名称命名，如 `dp-matrix_chain.cpp`

⑥ 数据文件建议以问题名称 + 数据源名称 + 序号命名，如 `matrix_chain-github-01.txt`

⑦ 实验结果文件建议以算法名称 + 数据源名称 + 序号 + 参数值命名，如 `dp-matrix_chain-01-8.txt`

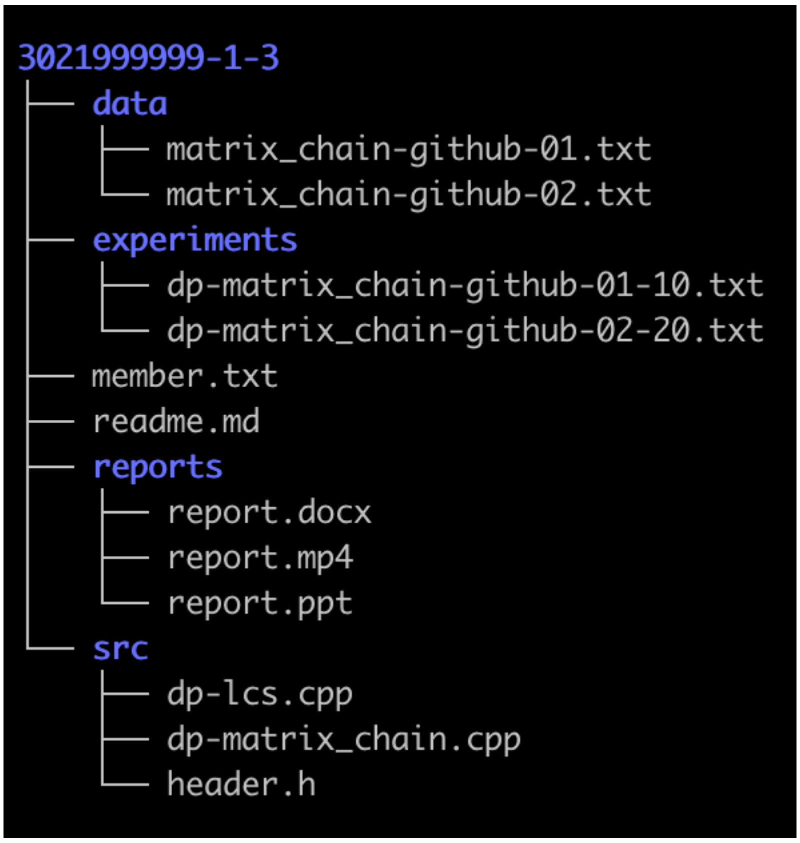
4.1 部分文档模版

- 文档目录结构，见图1
- readme.md

```
# 代码文件说明
包括文件名称、功能、输入的参数及格式、输出文件名称及格式
alg1.cpp

alg1.h
alg2.cpp
alg2.h

# 数据文件说明
包括文件名称、来源、格式及主要规模参数（如矩阵乘法链问题的矩阵个数 q 及维度向量等）
knapsack01_data1.txt

knapsack01_data2.txt

# 结果文件说明
包括输出的文件名称、使用的应用程序名称、调用的函数名称、参数设置及文件格式等。
exp_1.csv
exp_2.txt
```

- member.txt

```
学号   姓名   分工
3021999999 zhangsan dp 算法实现
```



```
3021999999-1-3
├── data
│   ├── matrix_chain-github-01.txt
│   └── matrix_chain-github-02.txt
├── experiments
│   ├── dp-matrix_chain-github-01-10.txt
│   └── dp-matrix_chain-github-02-20.txt
├── member.txt
├── readme.md
├── reports
│   ├── report.docx
│   ├── report.mp4
│   └── report.ppt
└── src
    ├── dp-lcs.cpp
    ├── dp-matrix_chain.cpp
    └── header.h
```

Figure 1: 递交文档目录结构

附件 1：问题求解数据集

- 0/1 包问题
  - JorikJooken(JJ) : 3240 instances. Link
  - Florida State University(FSU): 8 个数据集。Link
  - Unicauca University(UU): 31 个数据集。Link
  - David Pisinger(Pi): 3 个数据生成算法。link
- 子集和数问题
  - Florida State University(FSU): 7 个数据集。Link
- 矩阵乘法链问题
  - github: 9 个数据集。link
- 所有点对最短路径问题
  - gitlab : 利用 Ggen.cpp 随机生成。Link
- 最大集团问题
  - DIMACS-benchmark (DIMACS), 37 个数据集。Link
- TSP 问题
  - Florida State University(FSU): 6 个数据集。Link
  - TSP LIB(TSPLIB): 总计 143 个数据集。Link
- 最近点对问题
  - github: 4 个数据集。Link
- 数逆序对问题
  - github: 3 个数据集。Link

其中，数据源括号内为其缩写，用于定义数据文件

附件 2：文献阅读列表

1. Dynamic programming approaches for the traveling salesman problem with drone [1]

Abstract: A promising new delivery model involves the use of a delivery truck that collaborates with a drone to make deliveries. Effectively combining a truck and a drone gives rise to a new planning problem that is known as the traveling salesman problem with drone (TSP-D). This paper presents exact solution approaches for the TSP-D based on dynamic programming and provides an experimental comparison of these approaches. Our numerical experiments show that our approach can solve larger problems than the mathematical programming approaches that have been presented in the literature thus far. Moreover, we show that restrictions on the number of locations the truck can visit while the drone is away can help significantly reduce the solution times while having relatively little impact on the overall solution quality.

2. Divide and Conquer Dynamic Programming: An Almost Linear Time Change Point Detection Methodology in High Dimensions [2]

Abstract: We develop a novel, general and computationally efficient framework, called Divide and Conquer Dynamic Programming (DCDP), for localizing change points in time series data with high-dimensional features. DCDP deploys a class of greedy algorithms that are applicable to a broad variety of high-dimensional statistical models and can enjoy almost linear computational complexity. We investigate the performance of DCDP in three commonly studied change point settings in high dimensions: the mean model, the Gaussian graphical model, and the linear regression model. In all three cases, we derive non-asymptotic bounds for the accuracy of the DCDP change point estimators. We demonstrate that the DCDP procedures consistently estimate the change points with sharp, and in some cases, optimal rates while incurring significantly smaller computational costs than the best available algorithms. Our findings are supported by extensive numerical experiments on both synthetic and real data.

3. Features for the 0-1 knapsack problem based on inclusionwise maximal solutions [3]

Abstract: Decades of research on the 0-1 knapsack problem led to very efficient algorithms that are able to quickly solve large problem instances to optimality. This prompted researchers to also investigate whether relatively small problem instances exist that are hard for existing solvers and investigate which features characterize their hardness. Previously the authors proposed a new class of hard 0-1 knapsack problem instances and demonstrated that the properties of so-called inclusionwise maximal solutions (IMSs) can be important hardness indicators for this class. In the current paper, we formulate several new computationally challenging problems related to the IMSs of arbitrary 0-1 knapsack problem instances. Based on generalizations of previous work and new structural results about IMSs, we formulate polynomial and pseudopolynomial time algorithms for solving these problems. From this we derive a set of 14 computationally expensive features, which we calculate for two large datasets on a supercomputer in approximately 540 CPU-hours. We show that the proposed features contain important information related to the empirical hardness of a problem instance that was missing in earlier features from the literature by training machine learning models that can accurately predict the empirical hardness of a wide variety of 0-1 knapsack problem instances. Using the instance space analysis methodology, we also show that hard 0-1 knapsack problem instances are clustered together around a relatively dense region of the instance space and several features behave differently in the easy and hard parts of the instance space.

4. Faster algorithms for k-subset sum and variations [4]

Abstract: We present new, faster pseudopolynomial time algorithms for the k-Subset Sum problem, defined as follows: given a set Z of n positive integers and k targets $t_1, \cdots, t_k$, determine whether there exist k disjoint subsets $Z_1, \cdots, Z_k \subseteq Z$, such that $\sum(Z_i) = t_i$, for $i = 1, \cdots, k$. Assuming $t = \max\{t_1, \cdots, t_k\}$ is the maximum among the given targets, a standard dynamic programming approach based on Bellman' s algorithm can solve the problem in $O(nt^k)$ time. We build upon recent advances on Subset Sum due to Koiliaris and Xu, as well as Bringmann, in order to provide faster algorithms for k-Subset Sum. We devise two algorithms: a deterministic one of time complexity $\tilde{O}(n^{k/(k+1)}t^k)$ and a randomised one of $\tilde{O}(n+t^k)$ complexity. Additionally, we show how these algorithms can be modified in order to incorporate cardinality constraints enforced on the solution subsets. We further demonstrate how these algorithms can be used in order to cope with variations of k-Subset Sum, namely Subset Sum Ratio, k-Subset Sum Ratio and Multiple Subset Sum.

5. A branch-and-cut algorithm for the balanced traveling salesman problem [5]

Abstract: The balanced traveling salesman problem (BTSP) is a variant of the traveling salesman problem, in which one seeks a tour that minimizes the difference between the largest and smallest edge costs in the tour. The BTSP, which is obviously NP-hard, was first investigated by Larusic and Punnen (Comput Oper Res 38(5):868–875, 2011). They proposed several heuristics based on the double-threshold framework, which converge to good-quality solutions though not always optimal. In this paper, we design a special-purpose branch-and-cut algorithm for exactly solving the BTSP. In contrast with the classical TSP, due to the BTSP' s objective function, the efficiency of algorithms for solving the BTSP depends heavily on determining correctly the largest and smallest edge costs in the tour. In the proposed branch-and-cut algorithm, we develop several mechanisms based on local cutting planes, edge elimination, and variable fixing to locate those edge costs more precisely. Other critical ingredients in our method are algorithms for initializing lower and upper bounds on the optimal value of the BTSP, which serve as warm starts for the branch-and-cut algorithm. Experiments on the same testbed of TSPLIB instances show that our algorithm can solve 63 out of 65 instances to proven optimality.

6. Approximating binary longest common subsequence in almost-linear time [6]

Abstract: The Longest Common Subsequence (LCS) is a fundamental string similarity measure, and computing the LCS of two strings is a classic algorithms question. A textbook dynamic programming algorithm gives an exact algorithm in quadratic time, and this is essentially best possible under plausible fine-grained complexity assumptions, so a natural problem is to find faster approximation algorithms. When the inputs are two binary strings, there is a simple $\frac{1}{2}$−approximation in linear time: compute the longest common all-0s or all-1s subsequence. It has been open whether a better approximation is possible even in truly subquadratic time.

Rubinstein and Song showed that the answer is yes under the assumption that the two input strings have equal lengths. We settle the question, generalizing their result to unequal length strings, proving that, for any $\epsilon > 0$, there exists $\delta > 0$ and a $(\frac{1}{2} + \delta)$-approximation algorithm for binary LCS that runs in $n^{1+\epsilon}$ time. As a consequence of our result and a result of Akmal and Vassilevska-Williams, for any $\epsilon > 0$, there exists a $(\frac{1}{q} + \delta)$-approximation for LCS over q-ary strings in $n^{1+\epsilon}$ time. Our techniques build on the recent work of Guruswami, He, and Li who proved new bounds for error-correcting codes tolerating deletion errors. They prove a combinatorial "structure lemma" for strings which classifies them according to their oscillation patterns. We prove and use an algorithmic generalization of this structure lemma, which may be of independent interest.

References

[1] Paul Bouman, Niels Agatz, and Marie Schmidt. 2018. Dynamic programming approaches for the traveling salesman problem with drone [J]. *Networks* 72, (4) (December 2018), 528–542. 10.1002/net.21864

[2] Wanshan Li, Daren Wang, and Alessandro Rinaldo. 2023. Divide and conquer dynamic programming: An almost linear time change point detection methodology in high dimensions. Retrieved from https://arxiv.org/abs/2301.10942

[3] Jorik Jooken, Pieter Leyman, and Patrick De Causmaecker. 2022. Features for the 0-1 knapsack problem based on inclusionwise maximal solutions [J]. (2022), 1–30. Retrieved from http://arxiv.org/abs/2211.09665

[4] Antonis Antonopoulos, Aris Pagourtzis, Stavros Petsalakis, and ·Manolis Vasilakis. 2023. Faster algorithms for k-subset sum and variations [J]. *Journal of Combinatorial Optimization* 45, (2023), 24. 10.1007/s10878-022-00928-0

[5] Thi Quynh Trang Vo, Mourad Baiou, and Viet Hung Nguyen. 2024. A branch-and-cut algorithm for the balanced traveling salesman problem [J]. *J. Comb. Optim.* 47, (2) (January 2024). 10.1007/s10878-023-01097-4

[6] Xiaoyu He and Ray Li. 2023. Approximating binary longest common subsequence in almost-linear time. Retrieved from https://arxiv.org/abs/2211.16660