浙江大學
ZHEJIANG UNIVERSITY

姓名 _____

学号 _____

院所 _____

2025 年 11 月 30 日

Abstraction

Contents

1 实验与测试分析

本实验对三种算法方法（DFS+ 剪枝、动态规划（DP）、模拟退火（SA））在三分类问题上的性能进行了系统测试，测试设计考虑了 DFS 在小规模输入上的表现, 剪枝策略的优化, 输入规模 n 与总和 sum 的变化对 DP 算法性能的影响, 以及 SA 算法在大规模输入上的稳定性

1.1 测试数据设计

本实验共设计三类测试数据，分别用于评估 DFS、DP 与模拟退火（SA）算法在不同规模与结构下的性能表现。

首先，在 DFS 部分，我们设计了两类测试数据：一类是小规模 n 但具有较大 sum 的输入，用于考察剪枝策略在极端情况下的有效性；另一类是 n 与 sum 均较小的典型输入，用于基准测试 DFS 在常规规模下的运行性能。

对于 DP 算法，我们构造了三组测试数据。第一组固定 n 并逐步增大 sum，用于观察 DP 算法在状态空间随总和增大时的时间增长趋势。第二组固定 sum 但逐步增大 n，以分析 DP 对输入规模的敏感性。第三组则同时连续增加 n 与 sum，用于评估 DP 在大规模数据下的可计算范围，并对其经验时间复杂度进行拟合分析。

最后，在模拟退火（SA）部分，为了构造既能保证可解性、又能使 DP 难以处理的大规模测试实例 (符合 SA 的实际使用情况)，我们采用了一种基于“可解基”叠加的构造方法。具体而言，首先使用 DP 算法验证一批规模较小且均可分解的基准实例，记为 $S_1, S_2, \ldots, S_m$。从中任意选取 k 个作为构造基 (可重复选择)，记为 $A_1, A_2, \ldots, A_k$，并为每个基分配不同的权位，从而构造最终的测试集

$$T = \sum_{i=1}^{k} c^i \times A_i,$$

其中“$\times$”表示将集合 A_i 中的每个元素统一乘以权重 c^i。这种分层结构一方面保证了每个 A_i 的可解性在组合后仍然得以保留，使得整体实例 T 必然具有可行解；另一方面，由于恢复这些结构需要识别不同层次的数值模式，这一过程本身计算上并不容易，从而使得 T 成为某种意义上的“难实例”。基于此构造得到的数据集被用于评估 SA 算法在大规模复杂输入下的稳定性。

1.2 DFS + 剪枝测试结果

在小规模 n，大 sum 的数据下，DFS 算法在剪枝策略失效时，计算时间显著增加，部分数据甚至出现计算时间过长的现象。测试结果统计如下：

Table 1: DFS 算法运行时间统计（秒）

n	平均时间	最大时间	最小时间	方差
10	0.0080	0.0182	0.0054	5.12e-05
15	0.0081	0.0145	0.0044	2.19e-05
20	0.0078	0.0149	0.0042	1.92e-05
25	0.0064	0.0101	0.0043	6.82e-06
30	0.0921	0.1373	0.0049	0.0045
35	2.3341	9.2076	0.0043	7.41
40	3.1284	11.2849	0.0043	17.92
45	—	超时	0.0040	—
50	103.034	254.024	0.0044	12813.5

- 当剪枝条件失效时，DFS 的搜索空间呈指数级增长，时间复杂度接近 $O(k^n)$，其中 k 为每个元素可选分类数量。
- $n = 45$ 时出现超时现象，表明剪枝失效对 DFS 的影响极大。
- 对于小规模 $n < 30$，剪枝有效时算法运行时间在毫秒级，性能较好。

1.3 动态规划（DP）测试结果

为了系统评估 DP 算法在三分类问题上的性能，我们分别在以下三种典型场景下进行了实验：（1）固定 n、增大 sum；（2）固定 sum、增大 n；（3）同时逐步提高 n 与 sum。以下对三类实验结果分别进行分析。

1.3.1 固定 n，增大 sum 的情况

在该实验中，我们固定元素数量 n，逐步增加总和 sum，并记录 DP 算法的运行时间。实验结果如图 1 所示。可以明显观察到，随着 sum 的增大，DP 的运行时间呈现非线性增长趋势，但总体与 DP 的理论复杂度 $O(n \cdot sum)$ 保持一致。

从数据中可以看出：

- 当 sum 较小（例如 10^4 以下）时，运行时间增长相对平缓，均值维持在 $O(0.3 \sim 0.4)$ 秒的水平。
- 随着 sum 扩展到 10^5 量级，运行时间呈明显加速增长，尤其是方差也随之增大，显示状态表规模显著增加对性能的影响。
- 对不同复杂度假设的拟合结果显示，$O(n^2)$ 拟合曲线与实验数据最为贴合（MSE $= 0.772$），拟合参数为 $[4.662 \times 10^{-9}, -0.0684]$，远优于 $O(n)$（MSE $= 9.41$）或 $O(n \log n)$（MSE $= 7.45$）的拟合效果。

Table 2: 固定 n，增大 sum 时 DP 算法运行时间统计（秒）

$total\_sum$	mean	min	max	var
10002	0.3868	0.3765	0.4156	0.000266
20001	1.5310	1.4883	1.6145	0.003158
30000	3.3737	3.3543	3.4249	0.000853
40002	6.4288	6.3932	6.4482	0.000509
50001	11.5226	11.3661	11.6768	0.012789
60000	18.0380	16.6423	23.2785	8.589885
70002	24.0130	22.8261	26.7825	3.014150
80001	30.6717	29.9962	32.8286	1.466136
90000	37.7163	37.1789	39.1276	0.661494
100002	45.1194	43.9791	48.8319	4.401916

实验数据及拟合曲线见图 1。

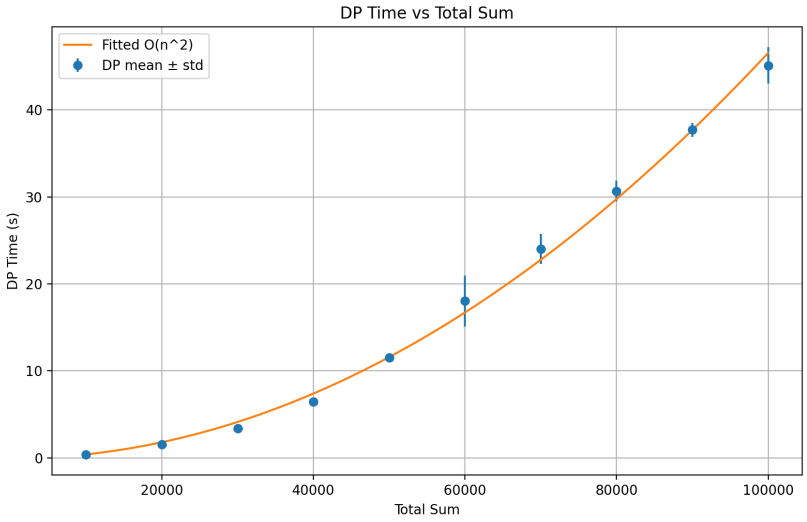

Figure 1: 固定 n，增大 sum 时 DP 算法运行时间及不同复杂度拟合曲线对比

综上，固定 n 条件下，DP 算法运行时间随 sum 单调增加，趋势稳定且可预测；即便在大规模 sum 条件下，其性能仍明显优于 DFS，表现出良好的稳定性和可扩展性。

1.3.2 固定 sum，增大 n 的情况

本组实验固定总和 sum，逐步增加输入规模 n，测试结果如图 2 所示。DP 的状态空间大小与 n 成线性关系，因此运行时间的增长主要由迭代次数增加所引起。

拟合曲线 2中可以观察到：

Table 3: 固定 sum，增大 n 时 DP 算法运行时间统计（秒）

n	mean	min	max	var
100	4.3085	4.2870	4.3247	0.000170
200	10.1568	9.9115	10.7473	0.090331
300	19.5679	15.8822	27.2218	16.318133
400	22.3316	21.8123	23.3965	0.314617
500	30.9510	28.7956	34.6313	4.063425
600	36.7893	35.6425	38.6712	1.007763
700	47.5941	43.3565	50.0603	5.547414
800	56.2361	49.5285	66.9230	38.305580
900	62.0149	58.8609	68.4820	12.228341
1000	68.6487	65.5686	76.3955	16.920649

- 在中小规模 n 范围内，DP 的运行时间基本保持线性增长，拟合结果表明 $O(n)$ 拟合较为贴合。通过最小二乘法拟合得到的时间复杂度函数为：

$$T(n) = 0.073154 \cdot n - 4.374624 \tag{1}$$

拟合的均方误差（MSE）为 12.146 仍然可以接受

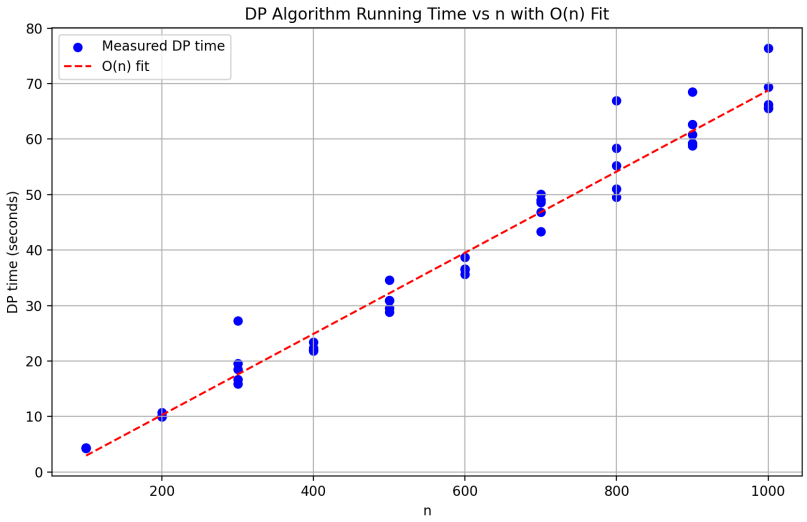

Figure 2: 固定 sum，增大 n 时 DP 算法运行时间及 O(n) 拟合曲线

1.3.3 同时增大 n 与 sum 的情况

为了进一步评估动态规划算法在大规模输入下的极限性能，我们系统地同时增大输入规模 n（项目数量）与 sum（目标和），使 DP 的状态空间呈二次方级增长（$O(n \cdot sum^2)$）。测试结果如图 3所示。

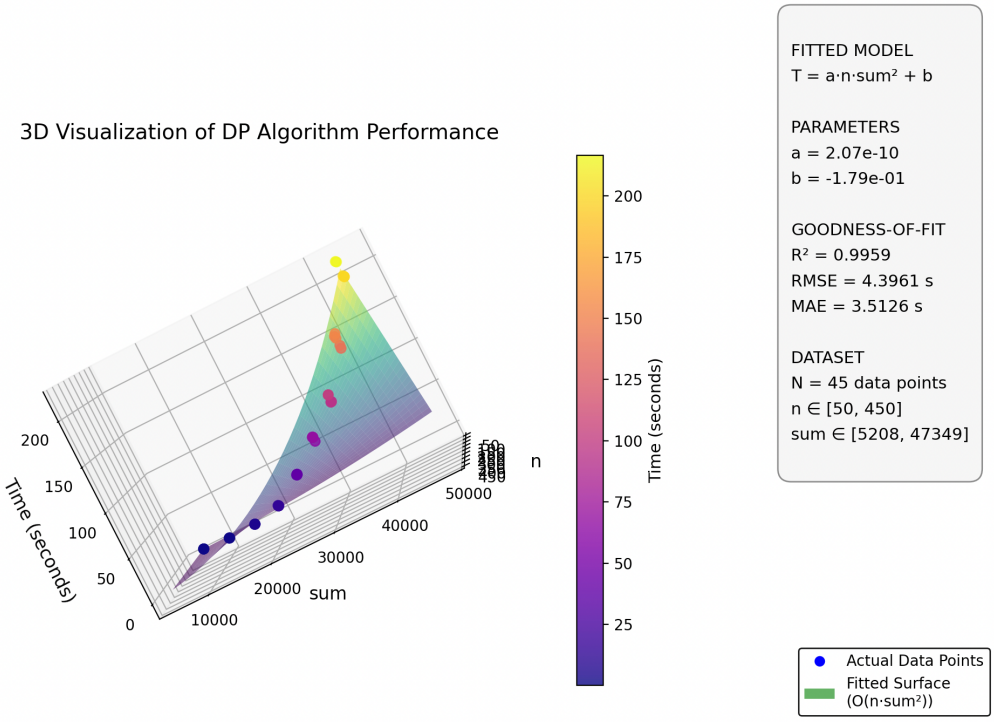

Figure 3: 动态规划算法在 n 与 sum 同时增大时的性能可视化。图中蓝色散点为实际运行时间，绿色曲面为基于 $O(n \cdot sum^2)$ 复杂度模型的拟合结果。参数拟合结果为 $T = 2.07 \times 10^{-10} \cdot n \cdot sum^2 - 0.179$，拟合优度 $R^2 = 0.9959$，表明模型能够准确描述算法的时间复杂度。

从可视化结果可以清晰观察到，当 n 与 sum 同时扩大时，DP 算法的运行时间呈快速增长趋势。具体而言：

- **指数级增长特征**：运行时间随 n 和 sum 的增大呈现明显的非线性增长，与 $O(n \cdot sum^2)$ 的理论复杂度高度一致（$R^2 = 0.9959$）。
- **硬件限制触及**：在 $n \in [50, 450]$，$sum \in [5208, 47349]$ 的测试范围内，最大运行时间已显著增长，并在部分数据点上观察到运行超时或内存溢出现象。
- **可计算性边界**：DP 算法的实际可计算范围严格受限于 $n \cdot sum^2$ 的规模，当两者同时增长时，算法很快到达不可计算区域，凸显了状态空间组合爆炸问题的本质性限制。

该实验从实证角度验证了模拟退火算法在大规模输入中具有不可替代的意义——动态规划算法虽然在小规模问题上具有最优性保证，但其计算复杂度使其难以扩展到现实世界的大规模优化问题，这强调了启发式算法在实际应用中的必要性。

1.4 模拟退火（SA）测试结果

对前述设计的 1000 个数据实例运行 SA 算法，其返回可行解的比例为 0.396

1.5 综合分析

- DFS+ 剪枝在小规模数据上效果显著，但在剪枝失效情况下，搜索空间指数增长，性能极差。
- DP 算法在中小规模数据上稳定高效，但受限于 n 和 sum 的乘积，超大数据可能无返回。
- SA 适用于大规模或近似求解场景，可快速得到可行解，但正确率低于 DP。

总体而言，不同算法适合不同规模和场景的三分类问题：DFS 更适合小规模精确求解，DP 适合中规模最优求解，SA 可用于大规模近似求解。