Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики» (СибГУТИ)

Кафедра ПМиК

Лабораторная работа 2
по дисциплине «Современные проблемы информатики»

Выполнил: ст. гр.
ЗМП-41 Лёвкин И. А.

Проверил: Лихачёв А.В.

Новосибирск 2025

Постановка задачи

Вариант 1.

1. Разработать компьютерную программу решения задачи о коммивояжере методом полного перебора.
2. Разработать компьютерную программу решения задачи о коммивояжере оптимизированным методом, указанным преподавателем.
3. Результаты работы представить в виде зависимости от числа пунктов маршрута отношения времени решения задачи методом полного перебора ко времени её решения оптимизированным методом.

Вариант 2.

1. Разработать компьютерную программу решения задачи Джонсона для двух станков методом полного перебора.
2. Разработать компьютерную программу решения задачи Джонсона для двух станков алгоритмом Джонсона.
3. Результаты работы представить в виде зависимости от числа деталей отношения времени решения задачи методом полного перебора ко времени её решения алгоритмом Джонсона.

Результаты:

Вариант 1.

Вывод программы:

```
Города: 4
  Полный перебор: 0.000020 сек
  Held-Karp (DP): 0.000028 сек
  Отношение времени перебор/динамическое программирование: 0.70

Города: 5
  Полный перебор: 0.000027 сек
  Held-Karp (DP): 0.000039 сек
  Отношение времени перебор/динамическое программирование: 0.68

Города: 6
  Полный перебор: 0.000122 сек
  Held-Karp (DP): 0.000072 сек
  Отношение времени перебор/динамическое программирование: 1.70

Города: 7
  Полный перебор: 0.000731 сек
  Held-Karp (DP): 0.000164 сек
  Отношение времени перебор/динамическое программирование: 4.46

Города: 8
  Полный перебор: 0.004574 сек
```

```
  Held-Karp (DP): 0.000525 сек
  Отношение времени перебор/динамическое программирование: 8.72

Города: 9
  Полный перебор: 0.036676 сек
  Held-Karp (DP): 0.000987 сек
  Отношение времени перебор/динамическое программирование: 37.15

Города: 10
  Полный перебор: 0.351454 сек
  Held-Karp (DP): 0.002693 сек
  Отношение времени перебор/динамическое программирование: 130.50

Города: 11
  Полный перебор: пропущен (слишком долго)
  Held-Karp (DP): 0.004528 сек

Города: 12
  Полный перебор: пропущен (слишком долго)
  Held-Karp (DP): 0.012136 сек

Города: 13
  Полный перебор: пропущен (слишком долго)
  Held-Karp (DP): 0.027218 сек

Города: 14
  Полный перебор: пропущен (слишком долго)
  Held-Karp (DP): 0.071902 сек

Города: 15
  Полный перебор: пропущен (слишком долго)
  Held-Karp (DP): 0.146836 сек

Города: 16
  Полный перебор: пропущен (слишком долго)
  Held-Karp (DP): 0.358237 сек
```

Рис. 1. Отношение метода перебора к методу динамического программирования

Вариант 2.

Вывод программы:

```
Деталей: 4
  Полный перебор: 0.000016 сек
  Алгоритм Джонсона: 0.000009 сек
  Отношение времени перебор/алгоритм Джонсона: 1.74

Деталей: 5
  Полный перебор: 0.000056 сек
  Алгоритм Джонсона: 0.000006 сек
  Отношение времени перебор/алгоритм Джонсона: 9.15

Деталей: 6
  Полный перебор: 0.000332 сек
  Алгоритм Джонсона: 0.000005 сек
  Отношение времени перебор/алгоритм Джонсона: 72.07

Деталей: 7
  Полный перебор: 0.002301 сек
  Алгоритм Джонсона: 0.000004 сек
  Отношение времени перебор/алгоритм Джонсона: 547.74

Деталей: 8
  Полный перебор: 0.019535 сек
  Алгоритм Джонсона: 0.000012 сек
  Отношение времени перебор/алгоритм Джонсона: 1627.95
```

```
Деталей: 9
  Полный перебор: 0.184292 сек
  Алгоритм Джонсона: 0.000010 сек
  Отношение времени перебор/алгоритм Джонсона: 17551.65

Деталей: 10
  Полный перебор: 1.970357 сек
  Алгоритм Джонсона: 0.000011 сек
  Отношение времени перебор/алгоритм Джонсона: 177509.60

Деталей: 11
  Полный перебор: 23.409337 сек
  Алгоритм Джонсона: 0.000014 сек
  Отношение времени перебор/алгоритм Джонсона: 1684125.05

Деталей: 12
  Полный перебор: пропущен (слишком долго)
  Алгоритм Джонсона: 0.000009 сек

Деталей: 13
  Полный перебор: пропущен (слишком долго)
  Алгоритм Джонсона: 0.000007 сек

Деталей: 14
  Полный перебор: пропущен (слишком долго)
  Алгоритм Джонсона: 0.000006 сек

Деталей: 15
  Полный перебор: пропущен (слишком долго)
  Алгоритм Джонсона: 0.000006 сек

Деталей: 16
  Полный перебор: пропущен (слишком долго)
  Алгоритм Джонсона: 0.000005 сек
```

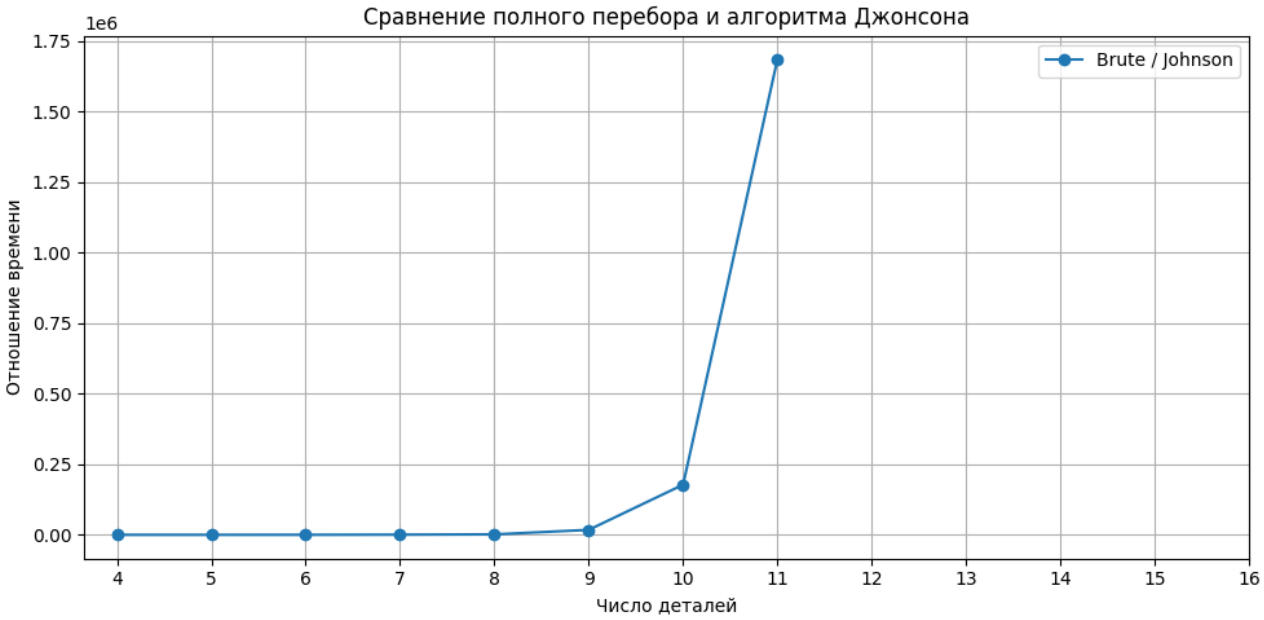

Рис. 2. Отношение метода перебора к алгоритму Джонсона

Листинг

tsp.py

```
import itertools
import time
import random
import matplotlib.pyplot as plt


def generate_distance_matrix(n, max_dist=100):
    return [
        [random.randint(1, max_dist) if i != j else 0 for j in range(n)]
        for i in range(n)
    ]


def calc_distance(route, dist_matrix):
    return (
        sum(dist_matrix[route[i]][route[i + 1]] for i in range(len(route)
- 1))
        + dist_matrix[route[-1]][route[0]]
    )


def tsp_brute_force(dist_matrix):
    n = len(dist_matrix)
    min_distance = float("inf")
    best_route = None

    for perm in itertools.permutations(range(1, n)):
```

```
        route = [0] + list(perm)
        dist = calc_distance(route, dist_matrix)
        if dist < min_distance:
            min_distance = dist
            best_route = route

    return best_route + [0], min_distance


def tsp_held_karp(dist_matrix):
    n = len(dist_matrix)
    memo = {}

    for k in range(1, n):
        memo[(1 << k, k)] = (dist_matrix[0][k], 0)

    for subset_size in range(2, n):
        for subset in itertools.combinations(range(1, n), subset_size):
            bits = sum(1 << k for k in subset)
            for k in subset:
                prev = bits & ~(1 << k)
                res = []
                for m in subset:
                    if m == k:
                        continue
                    res.append((memo[(prev, m)][0] + dist_matrix[m][k],
m))
                memo[(bits, k)] = min(res)

    bits = (1 << n) - 2
```

```
    res = [(memo[(bits, k)][0] + dist_matrix[k][0], k) for k in range(1,
n)]
    opt_cost, parent = min(res)

    path = [0]
    bits = (1 << n) - 2
    last = parent

    for _ in range(n - 1):
        path.append(last)
        new_bits = bits & ~(1 << last)
        _, last = memo[(bits, last)]
        bits = new_bits

    path.append(0)
    return path, opt_cost



def compare_algorithms():
    brute_times = []
    dp_times = []
    ratios = []
    cities_range = range(4, 17)

    for n in cities_range:
        matrix = generate_distance_matrix(n)

        print(f"\nГорода: {n}")

        if n <= 10:
```

```
            start = time.perf_counter()
            tsp_brute_force(matrix)
            brute_time = time.perf_counter() - start
            print(f"  Полный перебор: {brute_time:.6f} сек")
        else:
            brute_time = None
            print(f"  Полный перебор: пропущен (слишком долго)")

        start = time.perf_counter()
        tsp_held_karp(matrix)
        dp_time = time.perf_counter() - start
        print(f"  Held-Karp (DP): {dp_time:.6f} сек")

        brute_times.append(brute_time)
        dp_times.append(dp_time)

        if brute_time is not None:
            ratio = brute_time / dp_time if dp_time > 0 else float("inf")
            print(
                f"  Отношение времени перебор/динамическое
программирование: {ratio:.2f}"
            )
        else:
            ratio = None

        ratios.append(ratio)

    plt.figure(figsize=(10, 5))
    plt.plot(cities_range, ratios, marker="o", label="Ratio (Brute /
DP)")
```

```
    plt.xlabel("Число пунктов маршрута")
    plt.ylabel("Отношение времени (перебор / DP)")
    plt.title("Сравнение полного перебора и динамического
программирования (Held-Karp)")
    plt.grid(True)
    plt.xticks(cities_range)
    plt.legend()
    plt.tight_layout()
    plt.show()



if __name__ == "__main__":
    compare_algorithms()
```

johnson.py

```
import itertools
import time
import random
import matplotlib.pyplot as plt

# Вычисление времени выполнения последовательности
def calc_makespan(sequence, A, B):
    time_A = 0
    time_B = 0
    for i in sequence:
        time_A += A[i]
        time_B = max(time_A, time_B) + B[i]
    return time_B

def johnson_brute_force(A, B):
```

```
    n = len(A)
    best_seq = None
    best_time = float("inf")

    for perm in itertools.permutations(range(n)):
        t = calc_makespan(perm, A, B)
        if t < best_time:
            best_time = t
            best_seq = perm

    return list(best_seq), best_time

def johnson_algorithm(A, B):
    n = len(A)
    jobs = list(range(n))
    left = []
    right = []

    for job in jobs:
        if A[job] <= B[job]:
            left.append((A[job], job))
        else:
            right.append((B[job], job))

    left.sort()
    right.sort(reverse=True)

    sequence = [job for _, job in left] + [job for _, job in right]
    return sequence, calc_makespan(sequence, A, B)
```

```
# Генерация случайных данных
def generate_tasks(n, max_time=20):
    A = [random.randint(1, max_time) for _ in range(n)]
    B = [random.randint(1, max_time) for _ in range(n)]
    return A, B

def compare_johnson_algorithms():
    brute_times = []
    johnson_times = []
    ratios = []
    n_values = range(4, 17)

    for n in n_values:
        A, B = generate_tasks(n)

        print(f"\nДеталей: {n}")

        if n <= 11:
            start = time.perf_counter()
            johnson_brute_force(A, B)
            t_brute = time.perf_counter() - start
            print(f"  Полный перебор: {t_brute:.6f} сек")
        else:
            t_brute = None
            print(f"  Полный перебор: пропущен (слишком долго)")

        start = time.perf_counter()
        johnson_algorithm(A, B)
```

```
        t_johnson = time.perf_counter() - start
        print(f"  Алгоритм Джонсона: {t_johnson:.6f} сек")

        brute_times.append(t_brute)
        johnson_times.append(t_johnson)

        if t_brute is not None:
            ratio = t_brute / t_johnson if t_johnson > 0 else
float("inf")
            print(f"  Отношение времени перебор/алгоритм Джонсона:
{ratio:.2f}")
        else:
            ratio = None

        ratios.append(ratio)

    # Построение графика
    plt.figure(figsize=(10, 5))
    plt.plot(n_values, ratios, marker="o", label="Brute / Johnson")
    plt.xlabel("Число деталей")
    plt.ylabel("Отношение времени")
    plt.title("Сравнение полного перебора и алгоритма Джонсона")
    plt.grid(True)
    plt.xticks(n_values)
    plt.legend()
    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    compare_johnson_algorithms()
```