Smart shopping

with

search algorithims

presented by group 9

Tran Tue Khang - 11247303
Nguyen Khoi Nguyen - 11247331
Le Quang - 11247343
Nguyen Duc Huy - 11247297

Introduction

Real-world motivation & contributions

time-saving

saving fuel

less CO_2

Problems formulation

- **State** current market, set of items collected
- **Goal** all required items collected
- **Operators** move to a market selling any missing items; buy them
- **Operators** travel distance (Manhattan or Euclidean)

```
H(pos, collected):
    if collected already contains all required items: return 0
    max_d ← 0
    for each missing item k:
        d_k ← min over markets j that sell k of distance(pos, j)
        max_d ← max(max_d, d_k)
    return max_d
```

*H*euristic

Algorithms overview

Greedy-nearest

go to closest market with any missing item
→ fast, not optimal

GBFS

expand lowest ***h(n)***
→ can drift, ignores ***g(n)***

A*

expand lowest ***f(n) = g + h***
→ optimal with admissible ***h***

Method details

```
def greedy_nearest(home, required, markets, sells):
"""
Always move to the nearest market that sells at least one missing item.
markets: dict[name] -> (x,y)
sells:   dict[name] -> set(items)
"""
pos = markets[home]; collected=set(); path=[home]; cost=0
while collected != required:
    candidates = [(manhattan(pos, markets[j]), j)
                  for j in sells if (required - collected) & sells[j]]
    d, j = min(candidates)           # nearest useful market
    cost += d
    pos = markets[j]
    collected |= sells[j]
    path.append(j)
return path, cost
```

- Greedy-nearest

Method details

```
def gbfs(home, required, markets, sells):
start = (home, frozenset())
pq = [(H(markets[home], required, set(), markets, sells), 0, [home], start)]
seen = set()
while pq:
    h, g, path, (name, coll_f) = heappop(pq)
    if (name, coll_f) in seen:
        continue
    seen.add((name, coll_f))
    collected = set(coll_f)
    if collected == required:
        return path, g
    for j in sells:
        if (required - collected) & sells[j]:
            newC = collected | sells[j]
            ng = g + manhattan(markets[name], markets[j])
            nh = H(markets[j], required, newC, markets, sells)
            heappush(pq, (nh, ng, path+[j], (j, frozenset(newC))))
```

GBFS

Method details

```
            def astar(home, required, markets, sells):
start = (home, frozenset())
gscore = {start: 0}
pq = [(H(markets[home], required, set(), markets, sells), start, [home])]
seen=set()
while pq:
    f, (name, coll_f), path = heappop(pq)
    if (name, coll_f) in seen:
        continue
    seen.add((name, coll_f))
    collected = set(coll_f)
    g = gscore[(name, coll_f)]
    if collected == required:
        return path, g
    for j in sells:
        if (required - collected) & sells[j]:
            newC = collected | sells[j]
            ng = g + manhattan(markets[name], markets[j])
            s = (j, frozenset(newC))
            if ng < gscore.get(s, float('inf')):
                gscore[s] = ng
                nh = H(markets[j], required, newC, markets, sells)
                heappush(pq, (ng + nh, s, path+[j]))
```

A*

Assortments

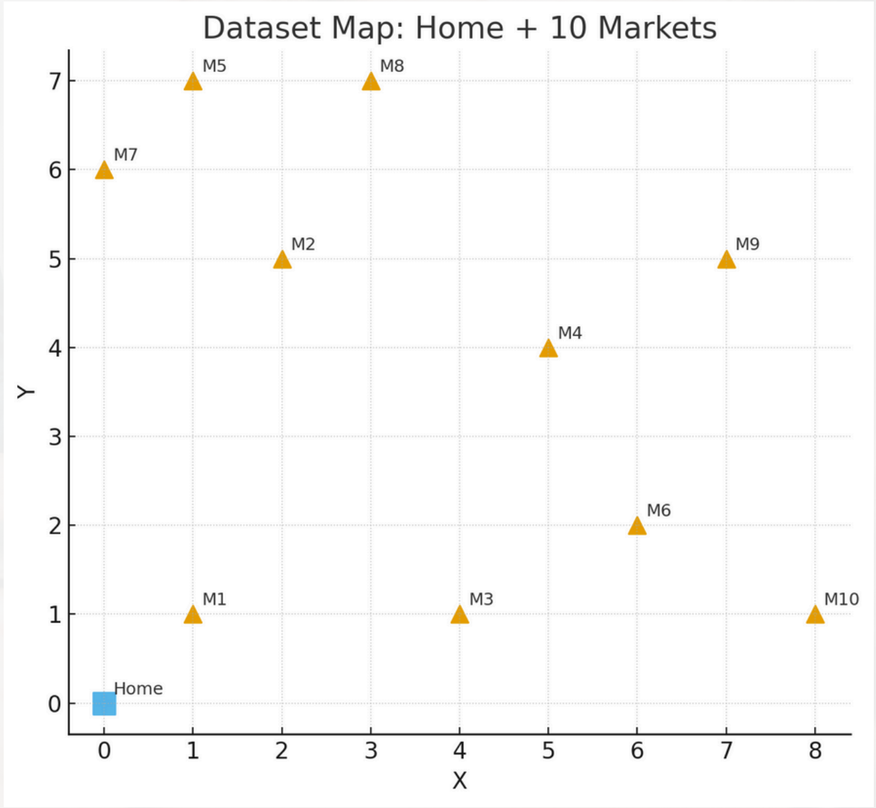
Each market sells a subset of 12 items.

Examples:

- M1: milk, eggs
- M4: fruit, vegetables, oil
- M7: vegetables, tofu, sugar
- M10: noodles, salt, spices

Shopping list

- Case A: 8 items
- Case B: 10 items
- Case C: 12 items

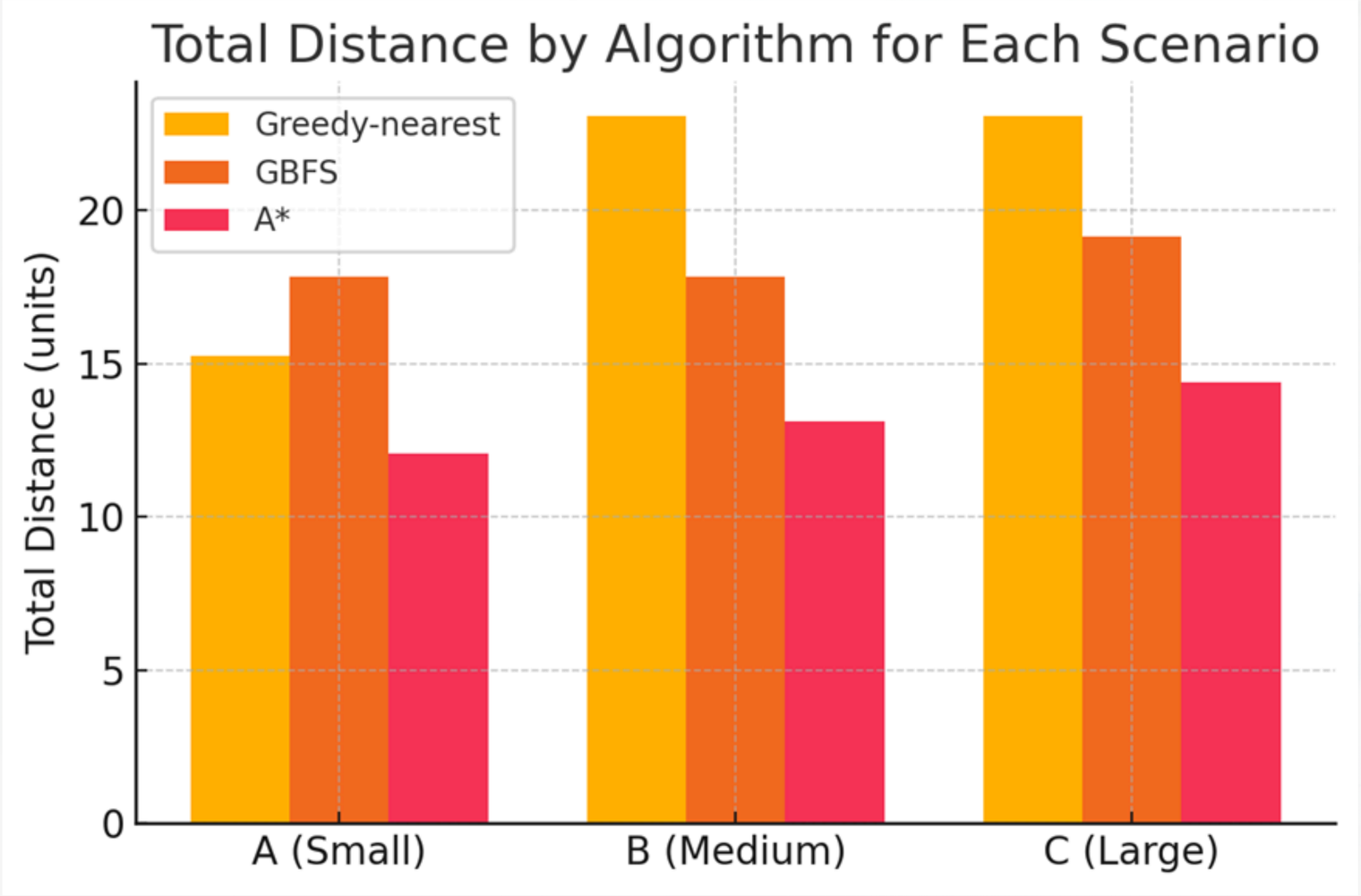


*D*ata & test cases

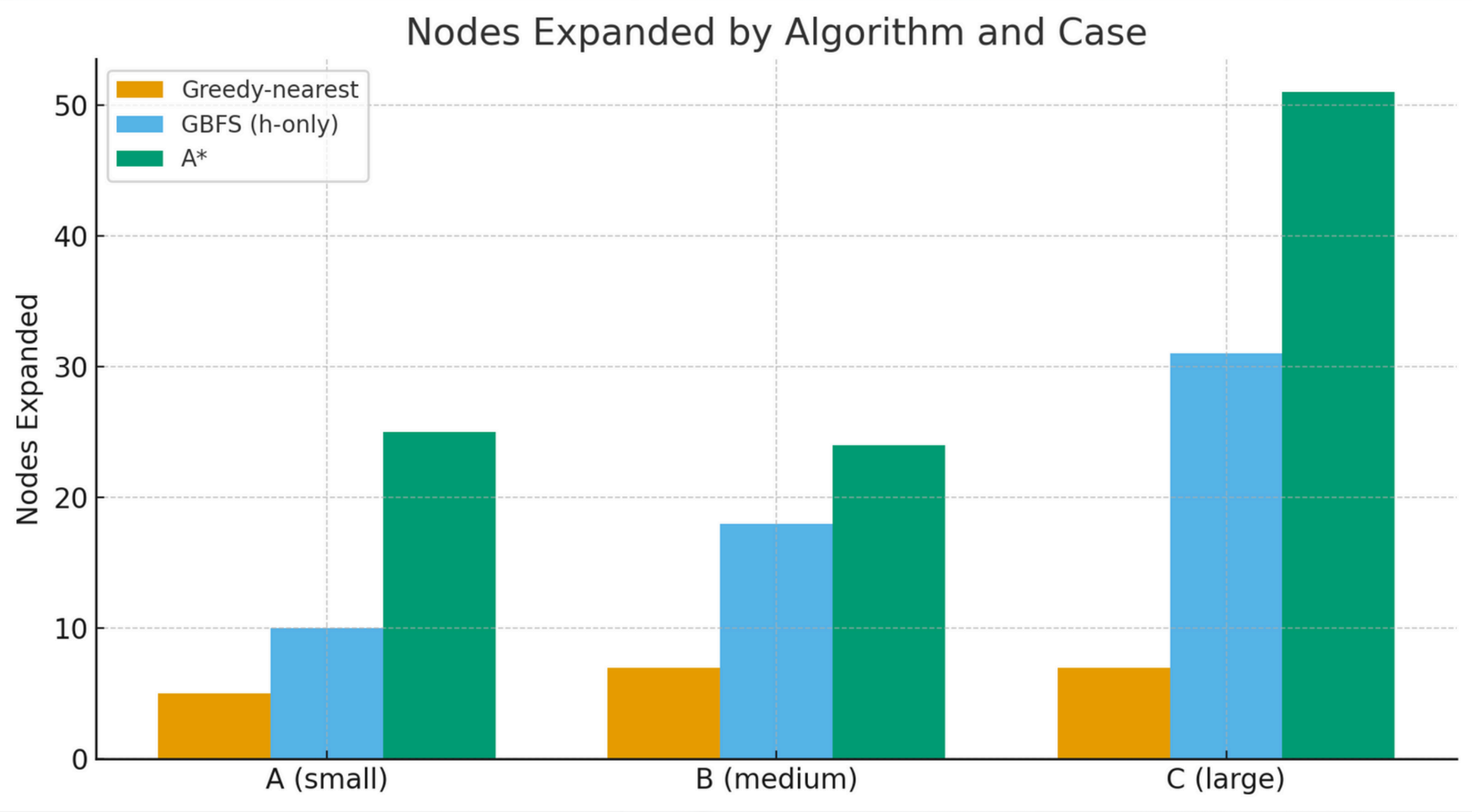
Results

Scenario	Algorithm	Total Distance (Cost)	Nodes Expanded (Depth)	Runtime (s)	Success	Route
A	Greedy-nearest	1.525	6	38	ĐÚNG	Home->M1->M3->M6->M4->M9->M10
	GBFS	1.783	87	3.781	ĐÚNG	Home->M3->M9->M8->M5->M2
	A*	1.205	80	3.175	ĐÚNG	Home->M1->M4->M3->M6->M10
B	Greedy-nearest	2.306	7	35	ĐÚNG	Home->M1->M3->M6->M4->M9->M10->M8
	GBFS	1.783	91	3.798	ĐÚNG	Home->M3->M9->M8->M5->M2
	A*	1.312	129	5.884	ĐÚNG	Home->M2->M1->M6->M8->M5
C	Greedy-nearest	2.306	7	33	ĐÚNG	Home->M1->M3->M6->M4->M9->M10->M8
	GBFS	1.916	78	4.094	ĐÚNG	Home->M3->M2->M8->M5->M9
	A*	1.439	203	9.925	ĐÚNG	Home->M1->M6->M2->M3->M4->M7->M5

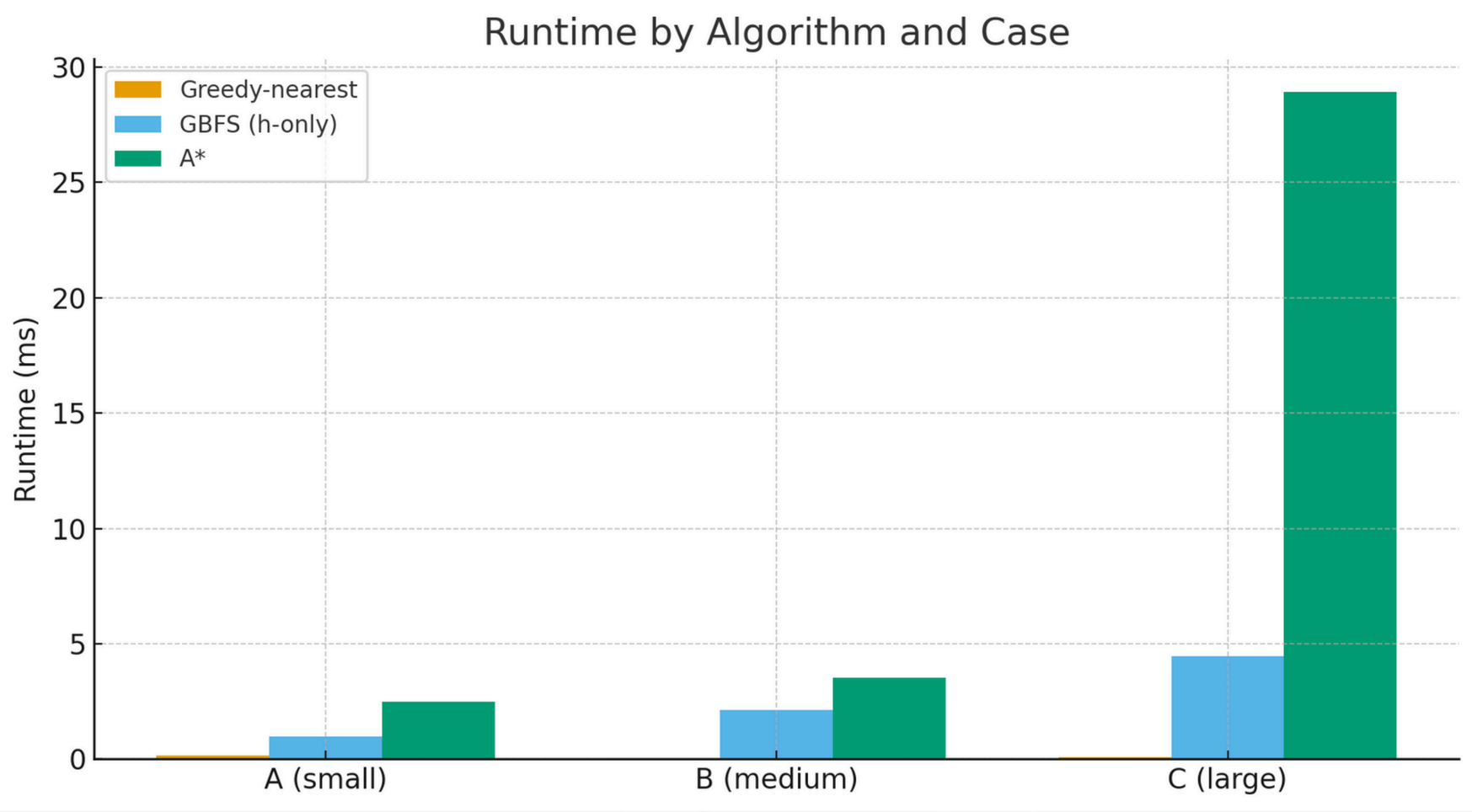
Results


Total Distance by Algorithm for Each Scenario
Greedy-nearest
GBFS
A*
Total Distance (units)
0
5
10
15
20
A (Small)
B (Medium)
C (Large)

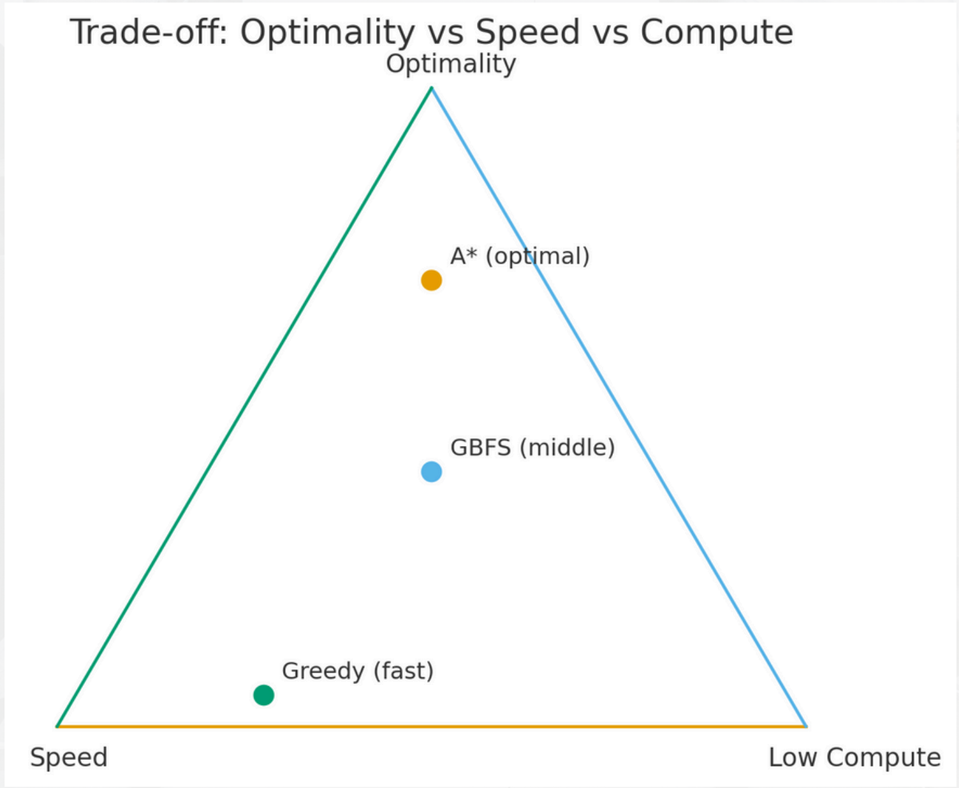
Results


Nodes Expanded by Algorithm and Case
Greedy-nearest
GBFS (h-only)
A*
Nodes Expanded
0
10
20
30
40
50
A (small)
B (medium)
C (large)

Results


Runtime by Algorithm and Case
Greedy-nearest
GBFS (h-only)
A*
Runtime (ms)
0
5
10
15
20
25
30
A (small)
B (medium)
C (large)

Dicussion, limitations & conclusion



The end.

Tran Tue Khang - 11247303
Nguyen Khoi Nguyen - 11247331
Le Quang - 11247343
Nguyen Duc Huy - 11247297