Name: _____

Today's Date: _____

Today's Goals

- Reason about the relative benefits of separate chaining and linear probing
- Compare theory to empirical results

Today's Question(s)

What's your favorite collision resolution strategy so far?

Lingering Questions

Naive Analysis of Linear Probing

Fact: If we succeed with probability p, the expected number of trials until success is $1/p$.

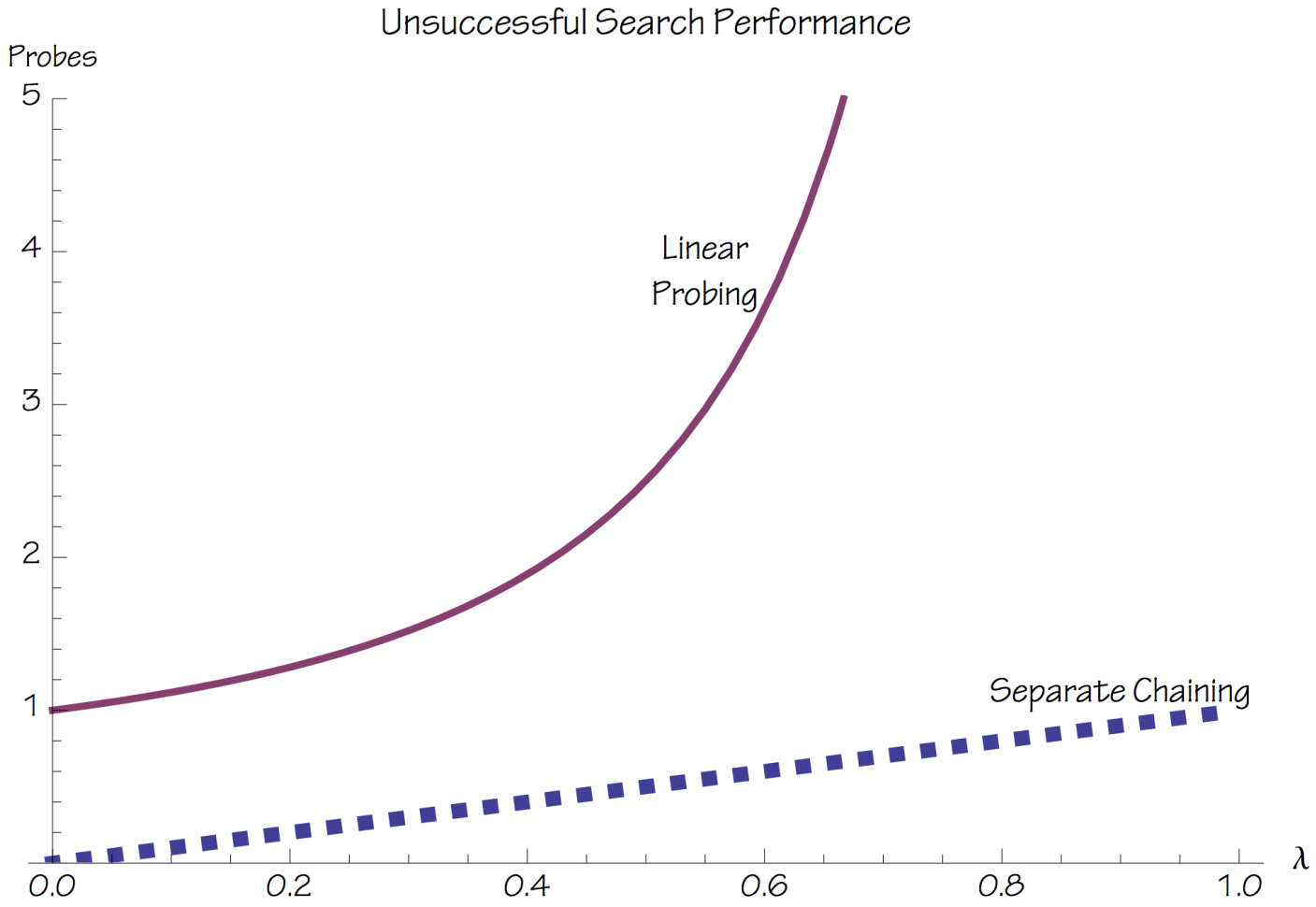
Maybe unsuccessful search checks $1/(1\text{-}\lambda)$ buckets?

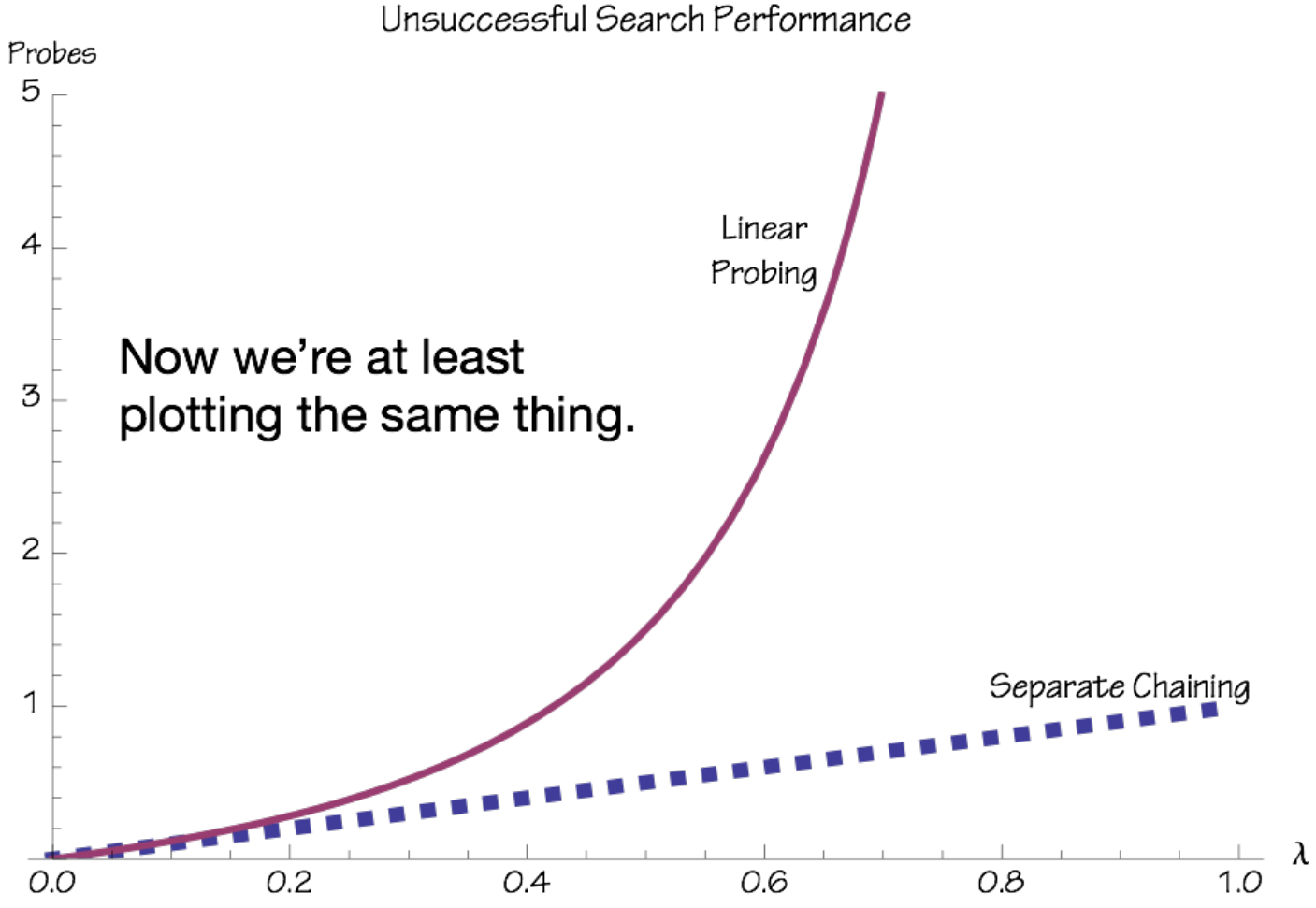
E.g. if λ=0.75, then

- $1/4$ of buckets are empty
- Expect to examine 4 buckets before stopping?

Lookup with Open Addressing

- Linear Probing:
  - Successful: $1 + 1/(1\text{-}\lambda)$
  - Unsuccessful: $1 + 1/(1\text{-}\lambda)\text{^}2$
- Quadratic Probing
  - Successful: $1\text{-}\ln(1\text{-}\lambda)\text{-}\lambda/2$
  - Unsuccessful: $1/(1\text{-}\lambda)$ - λ - $\ln(1\text{-}\lambda)$
- Double Hashing
  - Successful: $1/\lambda \ln(1/(1\text{-}\lambda))$
  - Unsuccessful: $\lambda/(1\text{-}\lambda) + 1 = 1/(1\text{-}\lambda)$


Unsuccessful Search Performance
Probes
5
4
3
2
1
Linear
Probing
Separate Chaining
0.0
0.2
0.4
0.6
0.8
1.0
λ

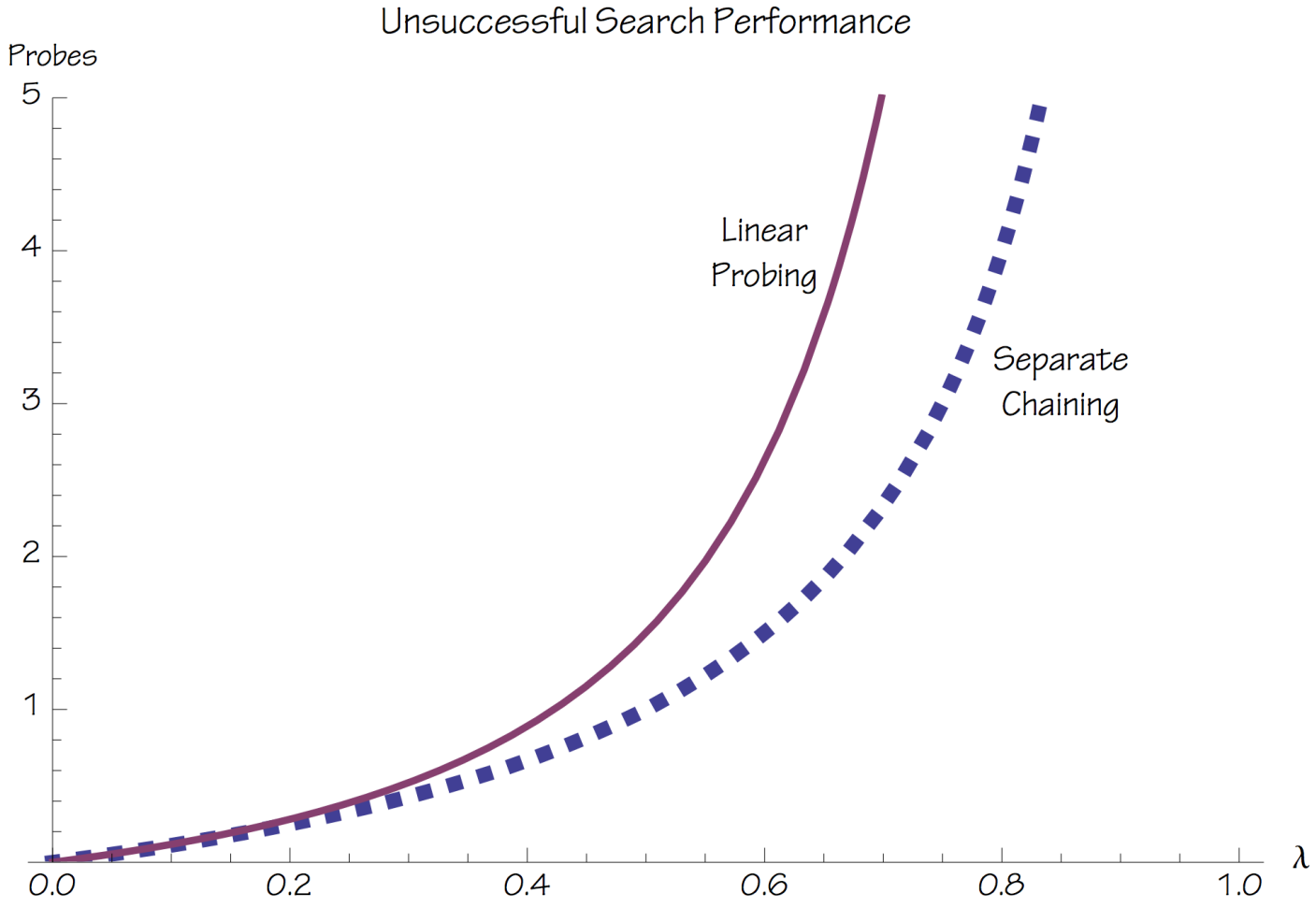

Unsuccessful Search Performance
Probes
5
4
3
2
1
Linear
Probing
Now we're at least
plotting the same thing.
Separate Chaining
0.0
0.2
0.4
0.6
0.8
1.0
λ

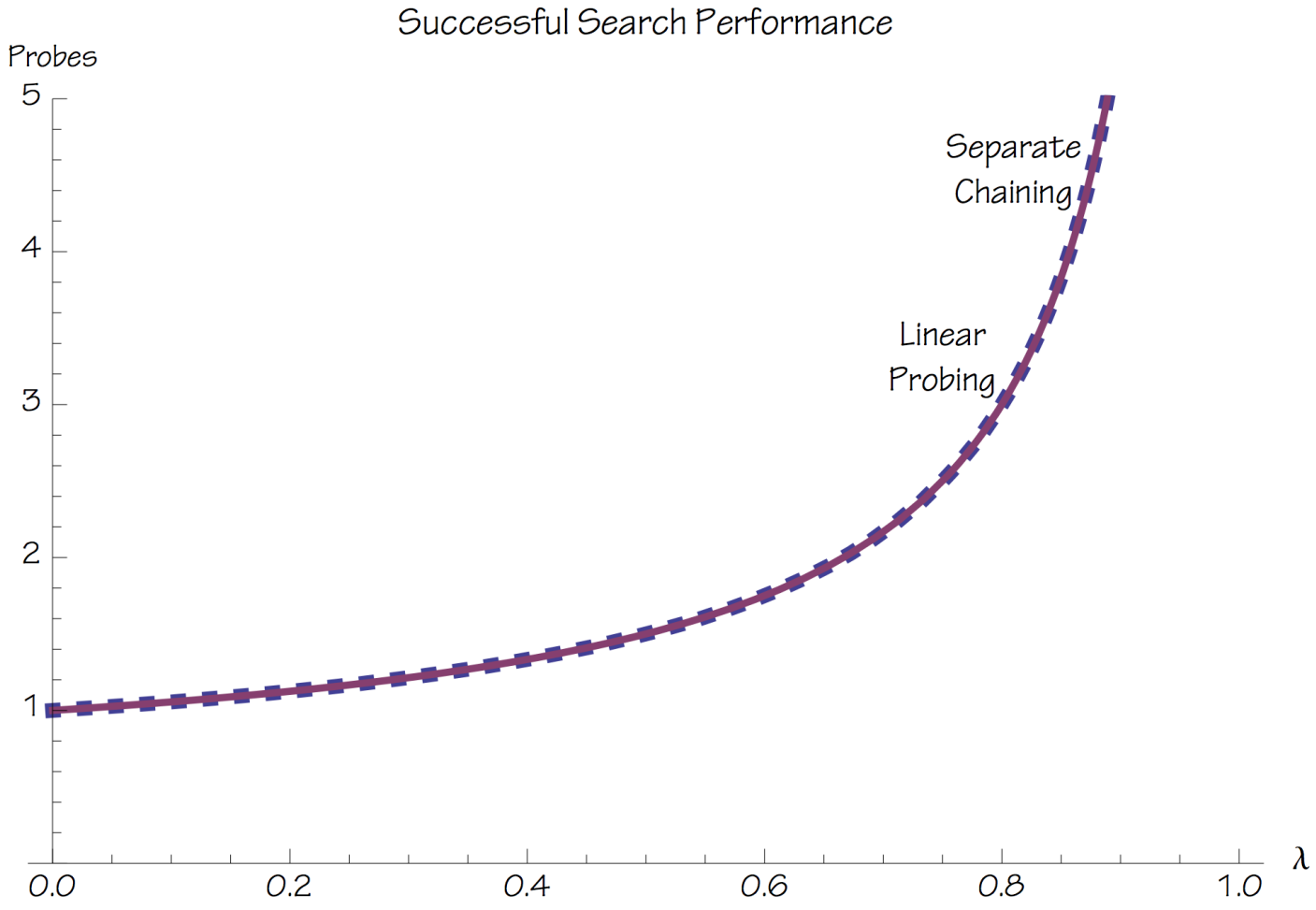
Comparing Separate Chaining to Open Addressing

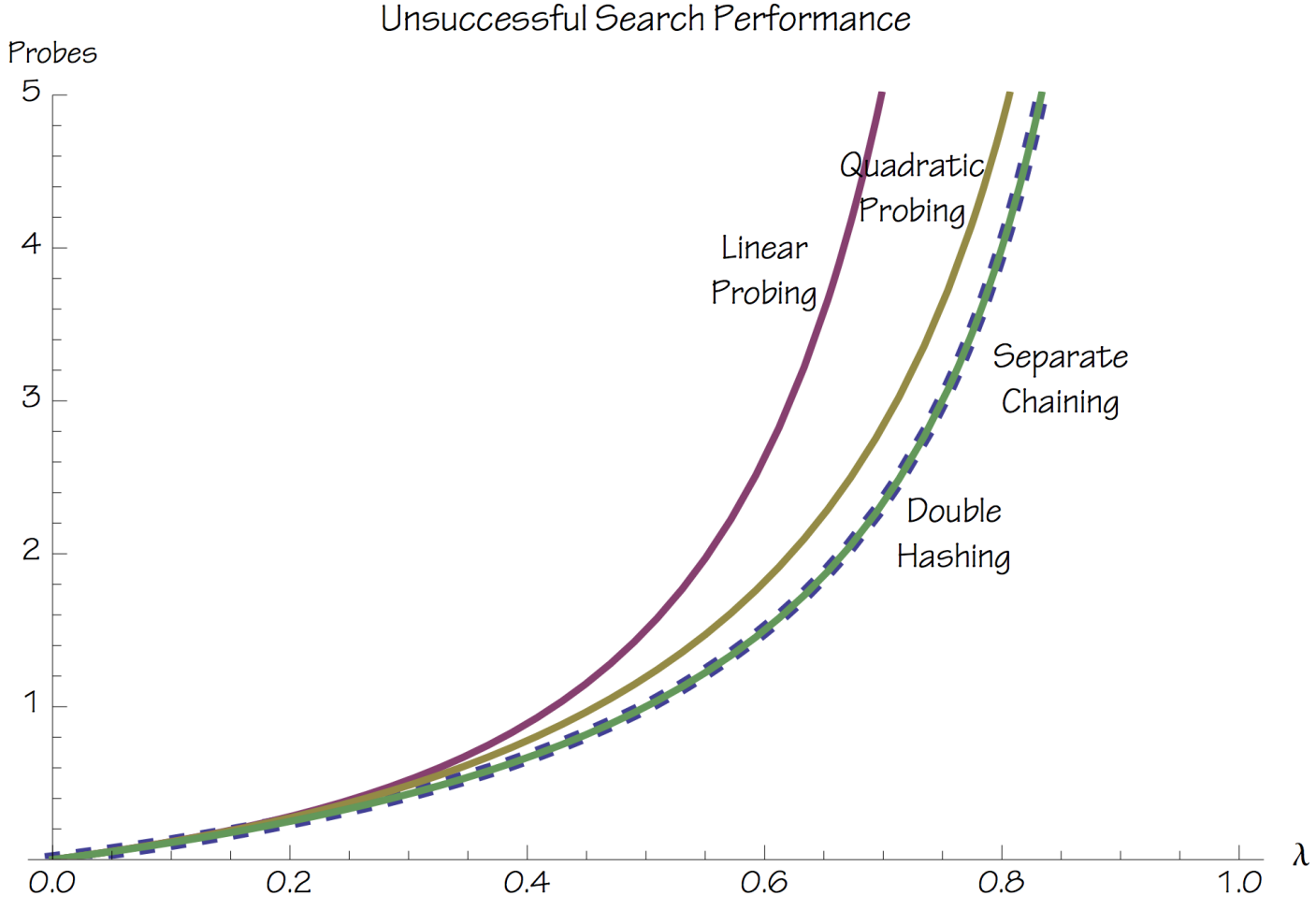
$$M_{oa} = M_{sc} + N$$

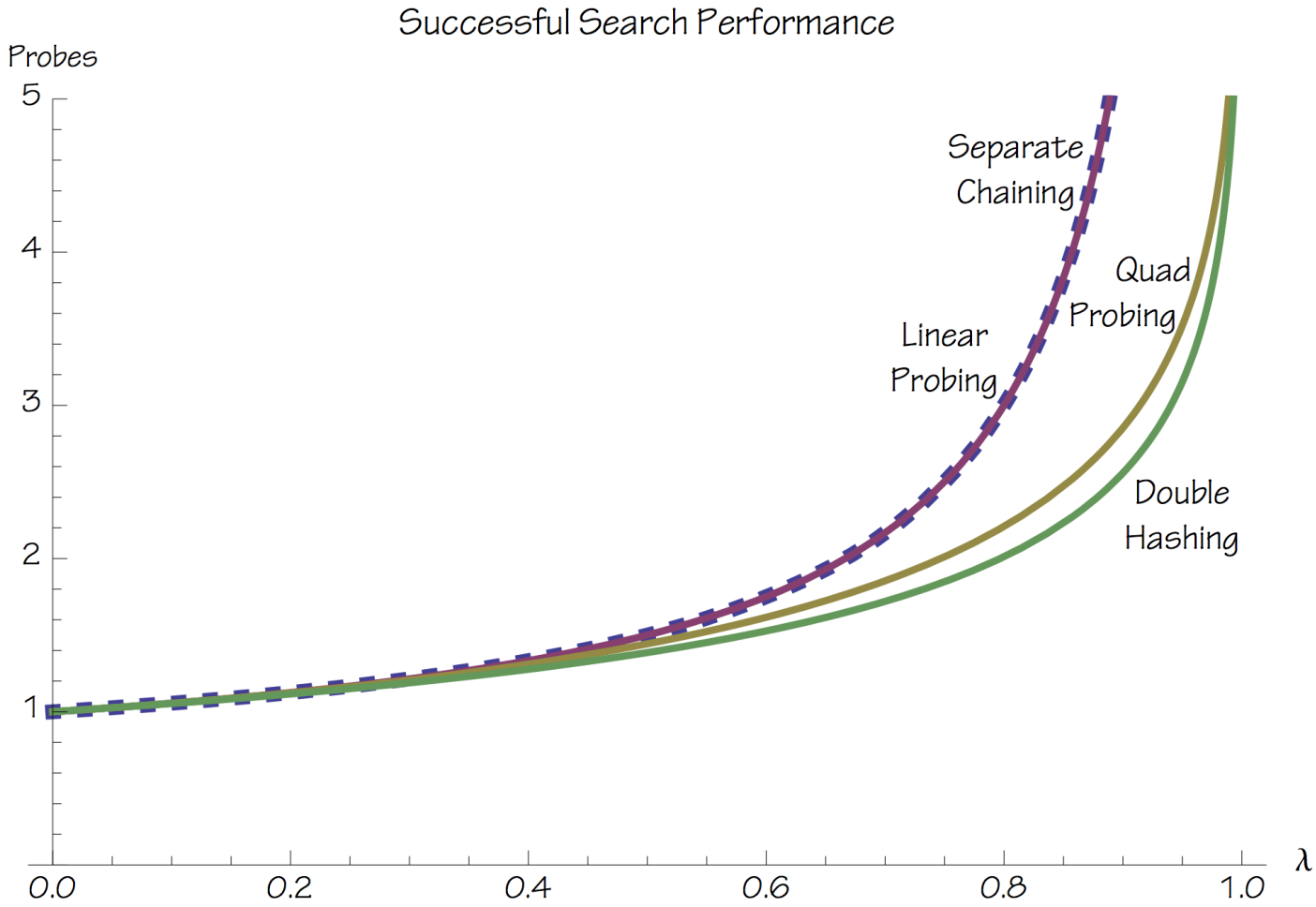
$$\lambda_{oa} = \frac{\lambda_{sc}}{1 + \lambda_{sc}}$$

$$\lambda sc = \frac{\lambda_{oa}}{1 - \lambda_{oa}}$$




Successful Search Performance
Probes
5
4
3
2
1
Separate
Chaining
Linear
Probing
0.0
0.2
0.4
0.6
0.8
1.0
λ


Unsuccessful Search Performance
Probes
5
4
3
2
1
Linear
Probing
Quadratic
Probing
Separate
Chaining
Double
Hashing
0.0
0.2
0.4
0.6
0.8
1.0
λ



Example Benchmark

Insert all the words in /home/student/data/web2 (234,936 words)

Look up all possible five letter "words"

- 11,881,386 total searches
- 8,494 successful searches
- 11,872,892 unsuccessful searches

Unsuccessful Search

Number of comparisons: best

λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	0	0	0	0
0.33	0.25	0	0	0	0
0.50	0.33	0	0	0	0
1.00	0.50	0	0	0	0
1.33	0.57	0	0	0	0
1.50	0.60	0	0	0	0
2.00	0.67	0	0	0	0
3.00	0.75	0	0	0	0
4.00	0.80	0	0	0	0

Number of comparisons: worst

λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	5	33	11	26
0.33	0.25	6	74	12	13
0.50	0.33	6	91	15	28
1.00	0.50	8	121	27	26
1.33	0.57	9	130	28	43
1.50	0.60	10	152	31	54
2.00	0.67	10	143	33	38
3.00	0.75	14	298	59	67
4.00	0.80	16	824	80	91

Number of comparisons: average

λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	0.25	0.29	0.27	0.25
0.33	0.25	0.33	0.40	0.37	0.34
0.50	0.33	0.50	0.65	0.56	0.51
1.00	0.50	1.00	1.62	1.16	1.01
1.33	0.57	1.33	2.36	1.56	1.35
1.50	0.60	1.50	2.94	1.77	1.52
2.00	0.67	2.00	4.37	2.36	2.03
3.00	0.75	3.00	8.93	3.65	3.09
4.00	0.80	4.00	16.07	4.84	4.13

Number of comparisons: expected

λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	0.25	0.28	0.27	0.25
0.33	0.25	0.33	0.39	0.37	0.33
0.50	0.33	0.50	0.63	0.57	0.50
1.00	0.50	1.00	1.50	1.19	1.00
1.33	0.57	1.33	2.22	1.61	1.33
1.50	0.60	1.50	2.62	1.82	1.50
2.00	0.67	2.00	4.00	2.43	2.00
3.00	0.75	3.00	7.50	3.64	3.00
4.00	0.80	4.00	11.99	4.81	4.00

Successful Search

Number of comparisons: best

λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	1	1	1	1
0.33	0.25	1	1	1	1
0.50	0.33	1	1	1	1
1.00	0.50	1	1	1	1
1.33	0.57	1	1	1	1
1.50	0.60	1	1	1	1
2.00	0.67	1	1	1	1
3.00	0.75	1	1	1	1
4.00	0.80	1	1	1	1

Number of comparisons: worst

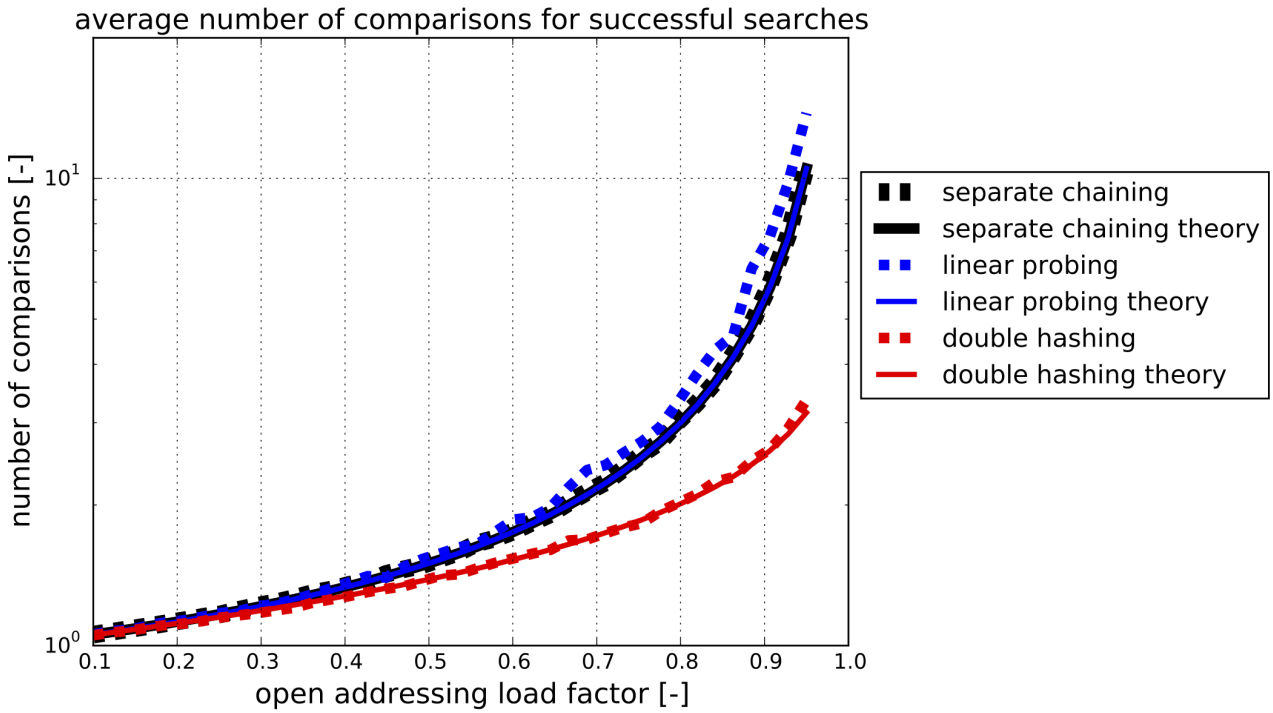
λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	5	25	11	7
0.33	0.25	5	47	9	8
0.50	0.33	6	65	11	10
1.00	0.50	8	99	17	15
1.33	0.57	8	94	17	17
1.50	0.60	10	115	20	18
2.00	0.67	10	117	24	24
3.00	0.75	14	230	48	33
4.00	0.80	14	761	48	89

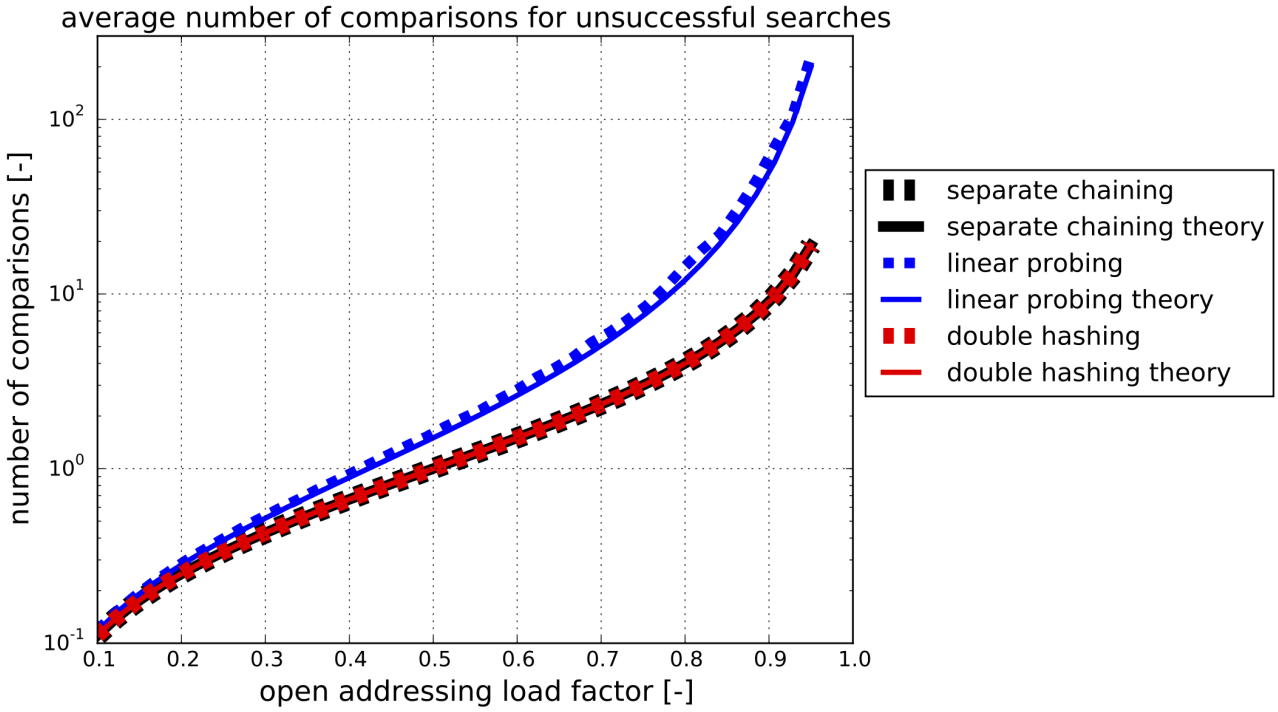
Number of comparisons: average

λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	1.12	1.13	1.12	1.12
0.33	0.25	1.17	1.17	1.16	1.16
0.50	0.33	1.25	1.26	1.24	1.22
1.00	0.50	1.50	1.54	1.44	1.40
1.33	0.57	1.67	1.71	1.55	1.49
1.50	0.60	1.75	1.83	1.61	1.55
2.00	0.67	2.00	2.08	1.75	1.67
3.00	0.75	2.50	2.75	2.00	1.89
4.00	0.80	3.00	3.42	2.20	2.06

Number of comparisons: expected

λ-sc	λ-oa	Sep Chain	Lin Prob	Quad Prob	Double Hash
0.25	0.20	1.13	1.13	1.12	1.12
0.33	0.25	1.17	1.17	1.16	1.15
0.50	0.33	1.25	1.25	1.24	1.22
1.00	0.50	1.50	1.50	1.44	1.39
1.33	0.57	1.67	1.67	1.56	1.48
1.50	0.60	1.75	1.75	1.62	1.53
2.00	0.67	2.00	2.00	1.77	1.65
3.00	0.75	2.50	2.50	2.01	1.85
4.00	0.80	3.00	3.00	2.21	2.01


average number of comparisons for successful searches
number of comparisons [-]
10^1
10^0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
open addressing load factor [-]
separate chaining
separate chaining theory
linear probing
linear probing theory
double hashing
double hashing theory


average number of comparisons for unsuccessful searches
number of comparisons [-]
10^2
10^1
10^0
10^{-1}
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
open addressing load factor [-]
separate chaining
separate chaining theory
linear probing
linear probing theory
double hashing
double hashing theory

Prime Table Sizes

Base	Prime Below	Prime Above
0	1	1
1	2	2
2	3	5
3	7	11
4	13	17
5	31	37
6	61	67
7	127	131
8	251	257
9	509	521
10	1021	1031
11	2039	2053
12	4093	4099

Base	Prime Below	Prime Above
13	8191	8209
14	16381	16411
15	32749	32771
16	65521	65537
17	131071	131101
18	262139	262147
19	524287	524309
20	1048573	1048583
21	2097143	2097169
22	4194301	4194319
23	8388593	8388617
24	16777213	16777259
25	33554393	33554467

Base	Prime Below	Prime Above
26	67108859	67108879
27	134217689	134217757
28	268435399	268435459
29	536870909	536870923
30	1073741789	1073741827
31	2147483647	2147483659
32	4294967291	4294967311
33	8589934583	8589934609
34	17179869143	17179869209
35	34359738337	34359738421
36	68719476731	68719476767
37	137438953447	137438953481
38	274877906899	274877906951

Base	Prime Below	Prime Above
39	549755813881	549755813911
40	1099511627689	1099511627791
41	2199023255531	2199023255579
42	4398046511093	4398046511119
43	8796093022151	8796093022237
44	17592186044399	17592186044423
45	35184372088777	35184372088891
46	70368744177643	70368744177679
47	140737488355213	140737488355333
48	281474976710597	281474976710677
49	562949953421231	562949953421381
50	1125899906842597	1125899906842679
51	2251799813685119	2251799813685269

Base	Prime Below	Prime Above
52	4503599627370449	4503599627370517
53	9007199254740881	9007199254740997
54	18014398509481951	18014398509482143
55	36028797018963913	36028797018963971
56	72057594037927931	72057594037928017
57	144115188075855859	144115188075855881
58	288230376151711717	288230376151711813
59	576460752303423433	576460752303423619
60	1152921504606846883	1152921504606847009
61	2305843009213693951	2305843009213693967
62	4611686018427387847	4611686018427388039
63	9223372036854775783	9223372036854775837
64	18446744073709551558	

```
uint64_t primeBelow(size_t power) {
    static const uint8_t offset[65] = {0,0,1,1,3,1,
           3,1,5,3,3,9,3,1,3,19,15,1,5,1,3,9,3,15,3,
           39,5,39,57,3,35,1,5,9,41,31,5,25,45,7,87,
           21,11,57,17,55,21,115,59,81,27,129,47,
           111,33,55,5,13,27,55,93,1,57,25,59};

    assert (power >= 0 && power <= 64);

    return (1LL << power) - offset[power];
}
```

```
uint64_t primeAbove(size_t power) {
    static const uint8_t offset[65] = {0,0,1,3,1,5,
           3,3,1,9,7,5,3,17,27,3,1,29,3,21,7,17,15,
           9,43,35,15,29,3,11,3,11,15,17,25,53,31,9,
           7,23,15,27,15,29,7,59,15,5,21,69,55,21,
           21,5,159,3,81,9,69,131,33,15,135,29};

    assert (power >= 0 && power <= 64);

    return (1LL << power) + offset[power];
}
```