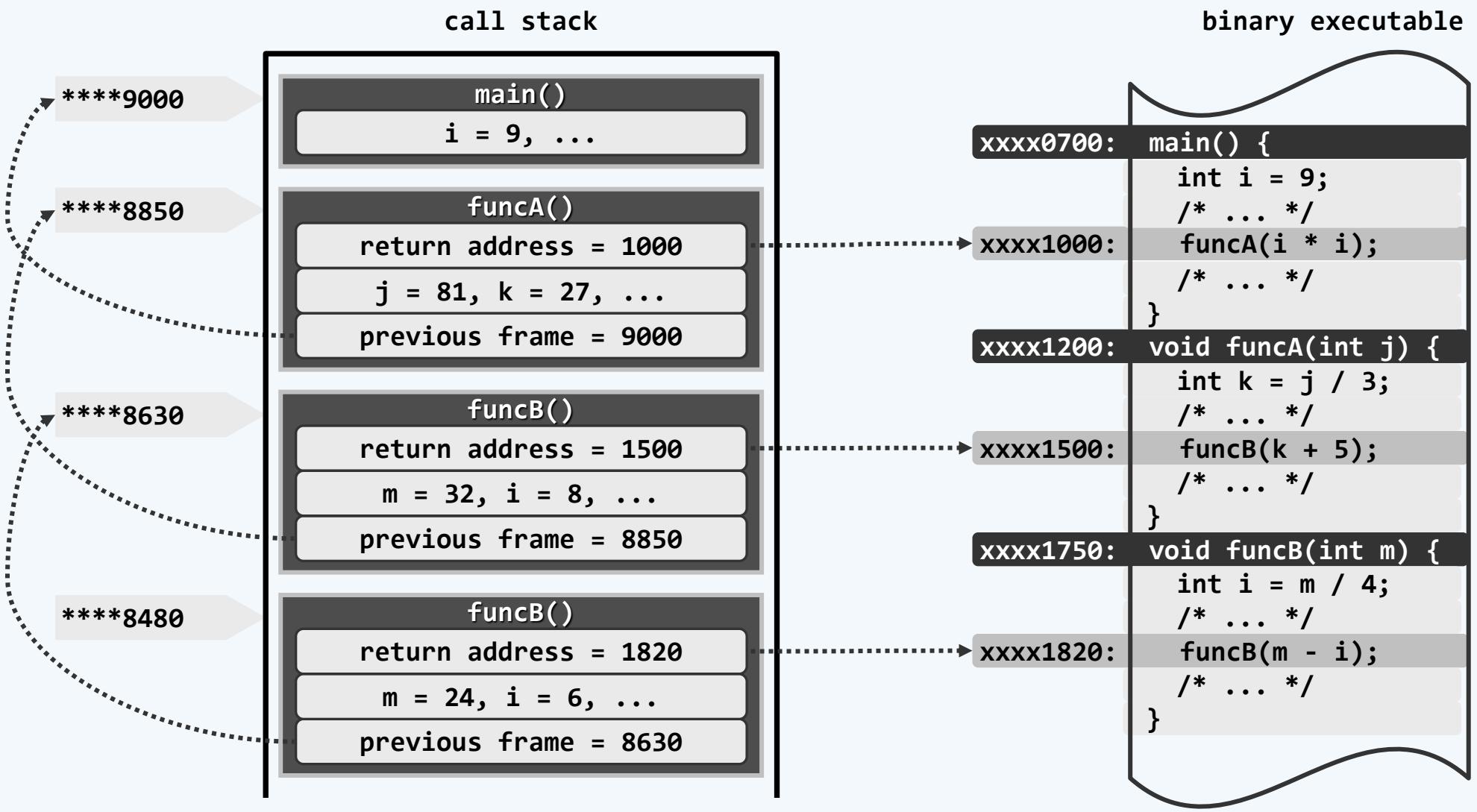
4.栈与队列

（b）栈与递归

邓俊辉

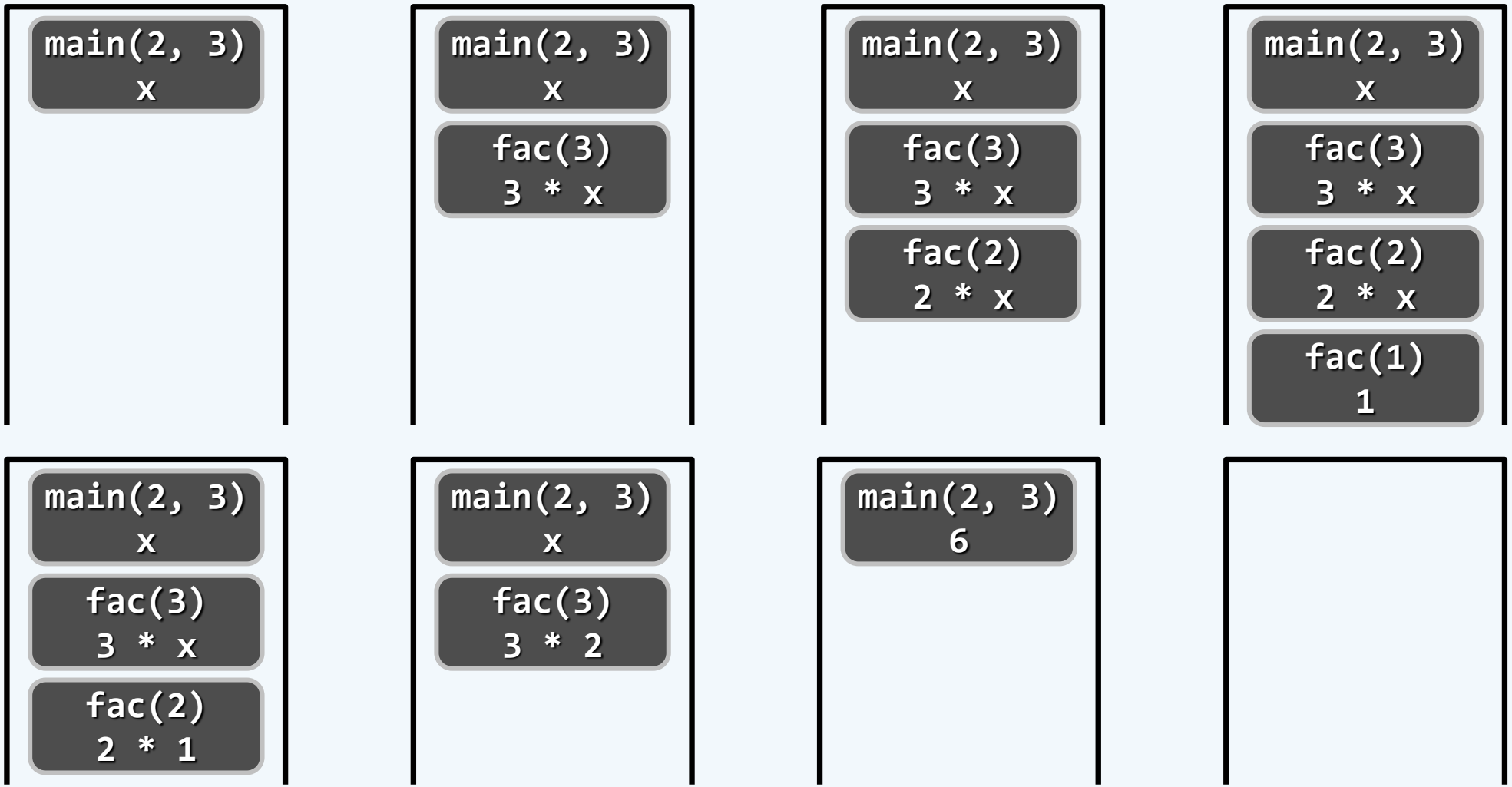
deng@tsinghua.edu.cn

函数调用栈


call stack
binary executable
****9000
main()
i = 9, ...
****8850
funcA()
return address = 1000
j = 81, k = 27, ...
previous frame = 9000
****8630
funcB()
return address = 1500
m = 32, i = 8, ...
previous frame = 8850
****8480
funcB()
return address = 1820
m = 24, i = 6, ...
previous frame = 8630
xxxx0700: main() {
int i = 9;
/* ... */
xxxx1000: funcA(i * i);
/* ... */
}
xxxx1200: void funcA(int j) {
int k = j / 3;
/* ... */
xxxx1500: funcB(k + 5);
/* ... */
}
xxxx1750: void funcB(int m) {
int i = m / 4;
/* ... */
xxxx1820: funcB(m - i);
/* ... */
}

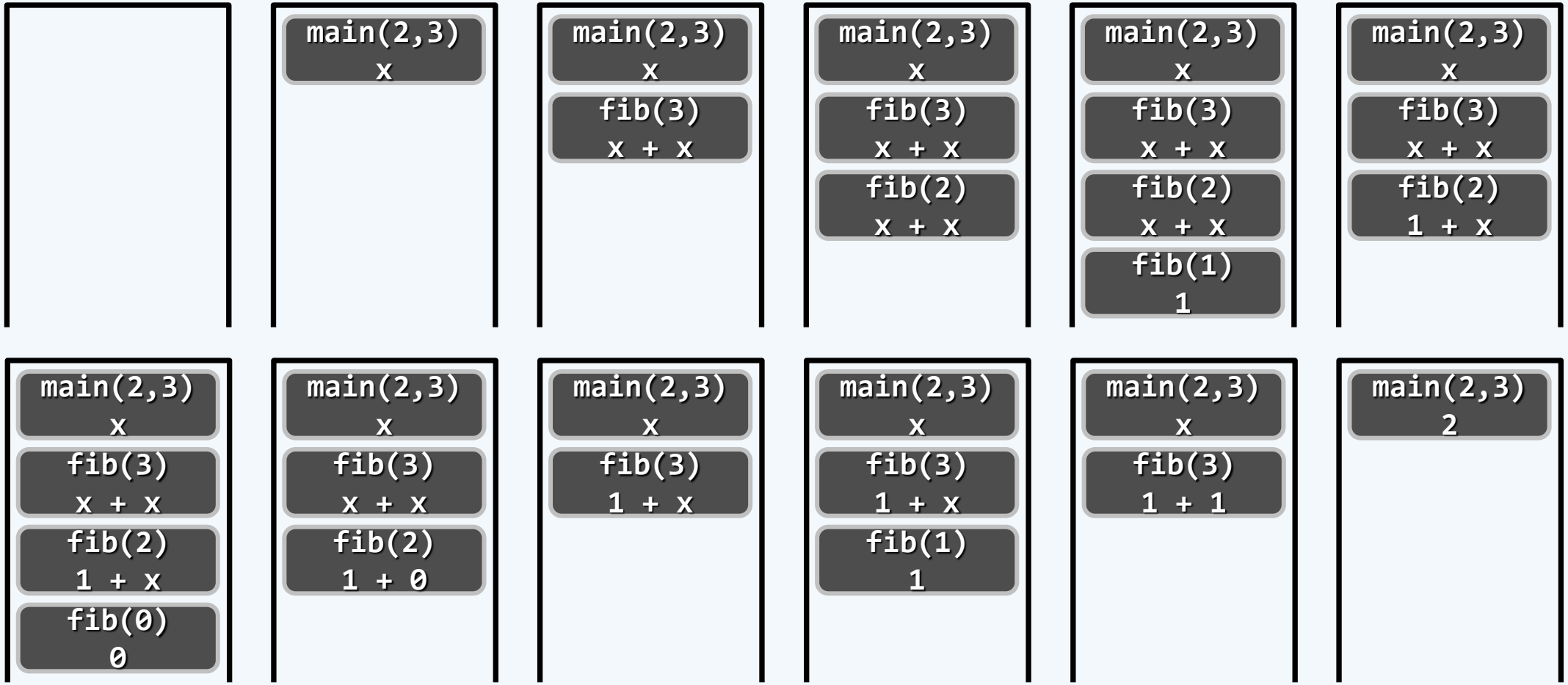
实例：fac()

❖ `int fac(int n) { return (n < 1) ? 1 : n * fac(n - 1); }`



实例：fib()

❖ `int fib(int n) { return (n < 2) ? n : fib(n - 1) + fib(n - 2); }`



实例：hailstone()

```
❖ hailstone(int n) {
      if (2 > n) return;
      n % 2 ? odd(n): even(n);
  }

❖ even(int n)
  { hailstone(n / 2); }
  odd(int n)
  { hailstone(3*n + 1); }

❖ main(int argc, char* argv[])
  { hailstone( atoi(argv[1]) ); }
```

call stack

main(2, 10)
hailstone(10)
even(10)
hailstone(5)
odd(5)
hailstone(16)
even(16)
hailstone(8)
even(8)
hailstone(4)
even(4)
hailstone(2)
even(2)
hailstone(1)

call stack

main(2, 27)
hailstone(27)
odd(27)
hailstone(82)
even(82)
hailstone(41)
odd(41)
hailstone(124)
even(124)
hailstone(62)
even(62)
hailstone(31)
odd(31)
hailstone(94)
... ...

避免递归

- 动机： 递归函数的空间复杂度，主要取决于最大递归深度，而非递归实例总数

  为隐式地维护调用栈，需花费额外的处理时间

- 方法： 将递归算法改写为迭代版本...

- `int fac(int n) { int f = 1; while (n > 1) f *= n--; return f; }` //O(1)空间

- `int fib(int n) { //O(1)空间`

```
    int f = 0, g = 1; while (0 < n--) { g += f; f = g - f; }
    return f;
}
```

- `void hailstone(int n) { while (1 < n) n = n % 2 ? 3*n + 1 : n/2; }` //O(1)空间

- 更为复杂的算法，其迭代版本往往需要显式地维护一个栈 //第五章