

June 16, 2016

## 1 Greedy Algorithms

Each step, choose the best local max. Fast, but proof of global optimum is often hard.

### 1.1 Disjoint Interval (Activity Selection)

Given a set of intervals, find largest set of disjoint intervals.

```
In [6]: # O(N log N)
def disjoint_1(intervals):
    intervals.sort(key=lambda (left, right):right)
    biggest_so_far = None
    answers = []
    for (l, r) in intervals:
        if biggest_so_far is None or biggest_so_far <= l:
            answers.append((l, r))
            biggest_so_far = r
    return answers

In [5]: assert disjoint_1([(1, 2), (3, 4), (2, 3), (1, 3)]) == [(1, 2), (2, 3), (3,
```

Proof of correctness: suppose we choose a sequence  $a_1 \dots a_n$ , but there's a longer length sequence  $b_1 \dots b_m, m > n$ . We know that  $b_{1R} > a_{1R}$  by the selection algorithm. Then  $a_1 b_2 \dots b_m$  is still optimal and disjoint. Then we compare  $a_2$  with  $b_2$ ,  $a_3$  with  $b_3$ , etc. But then  $b_{n+1}$  should be choosable by the selection algorithm, and thus a contradiction so we know our algorithm is optimal.