koko吃香蕉(leetcode 875)

```
class Solution {
public:
  int minEatingSpeed(std::vector<int>& piles, int h) {
    int left = 1, right = getMax(piles);

    while (left <= right) {
      int mid = left + (right - left) / 2;
      if (canFinish(piles, mid, h)) {
        right = mid - 1;
      } else {
        left = mid + 1;
      }
    }

    return left;
  }

private:
  bool canFinish(std::vector<int>& piles, int speed, int h) {
    int count = 0;
    for (auto item : piles) {
      count += time(item, speed);
    }

    return count <= h;
  }

  int time(int n, int speed) {
    return (n / speed) + (n % speed > 0 ? 1 : 0);
  }

  // 获取数组中最大值
  int getMax(std::vector<int>& piles) {
    int max = INT_MIN;
    for (auto item : piles) {
      max = (max > item ? max : item);
    }

    return max;
  }
};
```

货物运输(1011)

```
class Solution {
public:
  int shipWithinDays(std::vector<int>& weights, int days) {
```

```
    int left = getMax(weights), right = sum(weights);

    while (left <= right) {
      int mid = left + (right - left) / 2;
      if (canFinish(weights, mid, days)) {
        right = mid - 1;
      } else {
        left = mid + 1;
      }
    }

    return left;
  }

private:
  bool canFinish(std::vector<int>& weight, int speed, int days) {
    int j = 0;
    for (int i = 0; i < days; i++) {
      int maxcap = speed;
      while (0 <= (maxcap -= weight[j])) {
        j++;
        if (j == weight.size()) {
          return true;
        }
      }
    }
    return false;
  }

  int sum(std::vector<int>& weight) {
    int sum = 0;
    for (auto item : weight) {
      sum += item;
    }

    return sum;
  }

  int getMax(std::vector<int>& weights) {
    int max = INT_MIN;
    for (auto item : weights) {
      max = (max > item ? max : item);
    }

    return max;
  }
};
```