Палочки.

Заметим, что каждая палочка может входить в итоговый правильный k-угольник со стороной x не более, чем раз (делим целочисленно). Сторону x можно найти вещественным бинпоиском. Проверять можно ли составить k-угольник со стороной x можно пройдясь по массиву палочек, сложив сколько палочек со стороной x мы можем получить всего.

Бинпоиск будет работать потому что, если можно составить правильный k-угольник со стороной , тогда для стороны мы тоже сможем составить правильный k-угольник, для этого можно взять палочки для правильного k-угольника со стороной и уменьшить в размере до . Если мы не можем составить правильный k-угольник со стороной , тогда для стороны мы тоже не сможем составить правильный k-угольник. Т.к. количество подходящих палочек не увеличится. То есть вещественный бинпоиск будет правильно находить ответ, потому что чтобы получить максимальную площадь, нужно найти максимально большую подходящую сторону.

Псевдокод(Python 3 ☺):

1. **def** check(x):
2. cnt = 0
3. **for** i **in** range(n):
4. cnt += a[i] // x
5. **if** cnt >= k:
6. **return** True
7. **return** False

10. **def** bsearch():
11. left = 0
12. right = inf + 1
13. **for** i **in** range(100):
14. middle = (left + right) / 2
15. **if** check(middle):
16. left = middle
17. **else**:
18. right = middle
19. **return** left

22. #n - изначальное количество палочек; k - количество сторон k-угольника
23. n, k = map(int, input().split())
24. #a - массив длин палочек
25. a = list(map(float, input().split()))
26. inf = max(a)
27. **print**(bsearch())