Periodicity of sum-free sets

Write code to take a binary input decision sequence and use it to create a sum-free set of positive integers.

Investigate the output of two types of input sequences:

a) sequences with a finite string followed by all 1's

b) sequences consisting of a finite string repeated infinitely often: only compute finitely many terms of the sum-free set.

Questions to explore:

1. Are there input strings of the type above which appear to result in sets which are not ultimately periodic?

2. If so, what proportion of each of a) and b) appear to result in such sets?

Write your explorations up as a report, including code, appropriate pictures and conclusions.

```
def sum_free(x,n=0):
    if type(x) != type(''):
        if n==0: n = x.nbits()
        else: assert(n > x.nbits())
        x = bin(x)[2:].zfill(n)
    S = set()
    sums = set()
    next_int = 1
    for k in srange(len(x)):
        if x[k]=='1':
            S.add(next_int)
            sums |= {next_int+x for x in S}
        next_int += 1
        while (next_int in sums):
            next_int += 1
    return S
```

```
sum_free('0011')
```

```
{3, 4}
```

```
bin(2852)[2:]
```

```
'101100100100'
```

```
sum_free(2852)
```

```
{1, 4, 6, 13, 18}
```

```
bin(2852)[2:].zfill(16)
```

```
'0000101100100100'
```

```
sum_free(2852,16)
```

```
{5, 7, 8, 17, 20}
```

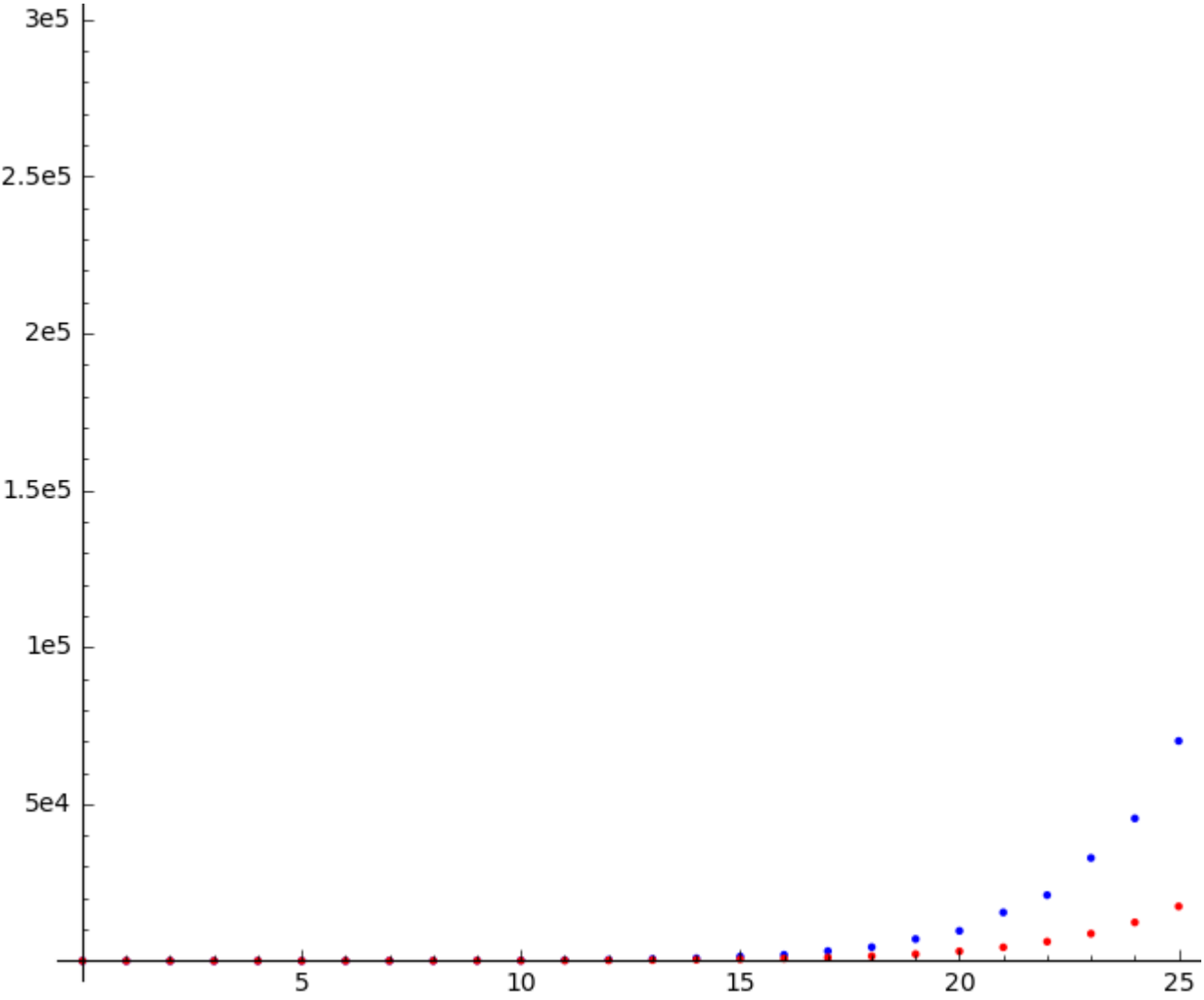
```
def is_sum_free(S):
    sums = set()
    for x in S:
        for y in S:
            sums.add(x+y)
    for s in S:
        if s in sums:
            return False
    return True

def extend(S,x):
    T = set(S)
    for subset in S:
        N = set(subset)
        N.add(x)
        if is_sum_free(N):
            T.add(frozenset(N))
    return T

def sf_subsets(n):
    if n==0:
        return set([frozenset()])
    return extend(sf_subsets(n-1),n)
```

```
R = [0 for i in srange(1,21)]
for x in srange(1,10):
    for y in srange(1,10):
        R[x+y] += 1
```

```
nsubsets = [len(sf_subsets(i)) for i in srange(30)]
A = list_plot(nsubsets)
```

```
P = [3*2^(i/2) for i in srange(30)]
B = list_plot(P, color='red')
```

```
show(A + B)
```

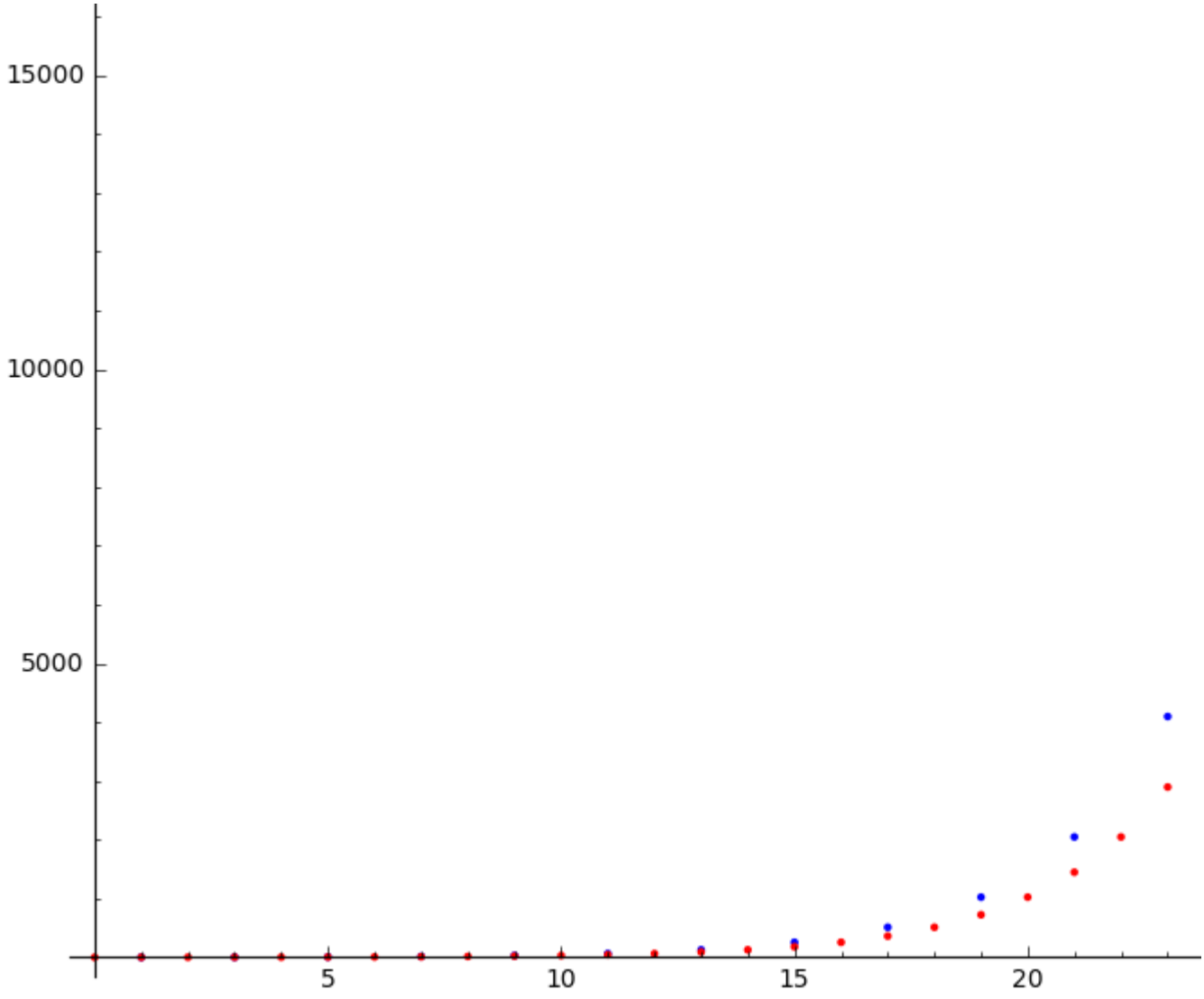



```
def sf_subsets_odd(n):
    assert(n%2==1)
    if n==1:
        return extend(set([frozenset()]),1)
    return extend(sf_subsets_odd(n-2),n)
```

```
nsubsets = [[2*i-1,len(sf_subsets_odd(2*i-1))] for i in srange(1,28)]
A = list_plot(nsubsets)
```

```
P = [2^(i/2) for i in srange(60)]
B = list_plot(P, color='red')
```

```
show(A + B)
```


15000
10000
5000
5
10
15
20