I decided to do separate analyses for the BRGC and the Knapsack algorithms. My specific solution only calls BRGC if n, the number of items, has changed. Otherwise, it reused the previously calculated powerset. I could just find the worst case, but I felt like this was an acceptable way to do it.

Analysis of BRGC Algorithm

1) Characterize Input: The number of powesets made depends on the integer n passed in the function.
2) The Basic Op: The basic operation is the four (4) insertions. It touches each element four times, so the basic operation executes 4n times on each recursive call. We just consider it to be n, though.
3) Same Count?: The same number of operations will be performed if n is the same.
4) Summation Exp.: A recurrence relation would be $C(n) = C(n-1) + n,\ n > 1,\ C(1) = 0$. It decreases n by one (1) on each recursive call, and then does n work when it returns from the recursion.
5) Closed Form: We can't use the master theorem, but we can easily solve it with backwards substitution. Solving gives $n^2 - n$, or simply n^2.
6) Growth Function: $n^2 - n \in O(n^2)$, or quadratic

Analysis of Knapsack Algorithm

1) Characterize Input: Input size is based on how many elements are in the powerset list and the size of those elements. The element size is based on the vector size.
2) The Basic Op: The basic operation is the comparison, specifically $if\ (str[i] == '1')$.
3) Same Count?: The same number of operations will execute as long as the size of the powerset is the same.
4) Summation Exp.: A summation expression that characterizes the number of times the basic operation executes would be $\sum_{i=1}^{n} \sum_{j=1}^{log_2 n} 1$. The basic op operates $log_2 n$ times for each n in the outer loop.
5) Closed Form: Finding a closed form gives $n\ log_2 n$.
6) Growth Function: $n\ log_2 n \in O(n\ log_2 n)$