Super-Duper Concentrators and the Pebble Game

Note: For this paper, the section on superconcentrators primarily follows Chapter 23 of the translated version of Gems of Theoretical Computer Science *[SP98]. The section on super-duper concentrators primarily follows Chapter 24 of the same book [SP98], with the exception that the proof of the main lemma (Lemma 3.9) follows the proof in the original paper [PTC76], which I believed to provide more intuition for the lemma. Moreover, the construction of the actual super-duper concentrator in the book [SP98] and in the paper [PTC76] were slightly different, and I used a construction midway between the two, which I thought was most convenient for making the proofs that followed as clear as possible.*

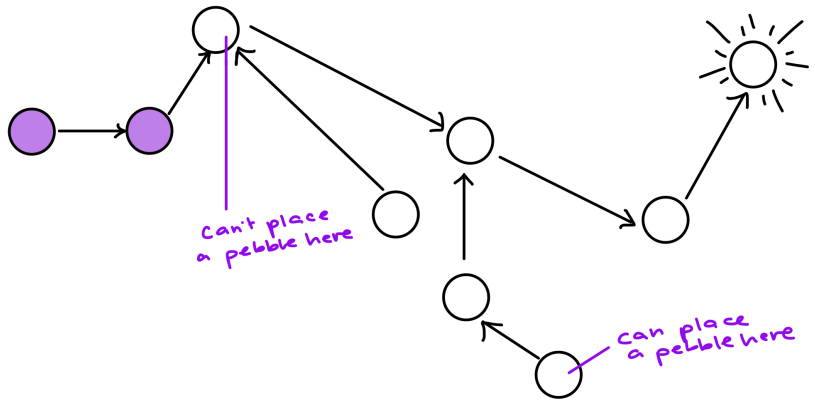
1 Introduction and Background

The pebble game, like most games useful for theoretical analysis, is a fun game to play if you have a very strange definition of fun. However, it is a very cute and interesting way to study properties of graphs, and as we showed in class, we can use the pebble game as a metaphor for the computations made by an algorithm in order to further understand limits on their space complexity. In this paper, we will explore some very strange and interesting graphs, which are hard to pebble, in order to show a lower bound on the number of pebbles needed to win the general pebble game.

Reminder 1.1. As we learned in class, the **pebble game** is a game defined on a directed acyclic graph. Our goal is to get a pebble on a given sink vertex, while using as few total pebbles as possible.

To accomplish this goal, we have the following rules:

1. We define a **predecessor** of a vertex j to be any vertex i such that there is a directed edge $i \to j$. Then, we can only place a pebble on j if there are currently pebbles on all of j's predecessors.
2. As the above implies, we can always place a pebble on a source vertex (one with no in edges).
3. At any time, we can remove a pebble from any vertex with no cost.



Moreover, in class we proved the following theorem:

Theorem 1.2 (Hopcroft-Paul-Valiant [HPV77])**.** Every v-vertex constant in-degree graph can be pebbled with $v/\log v$ pebbles.

That is, in class we found a strategy to pebble every v-vertex constant in-degree graph with $v/\log v$ pebbles. We will show in the rest of this paper that this bound is tight; that is, we will find a family of graphs such that we cannot win the pebble game with less than $\Omega(v/\log v)$ pebbles.

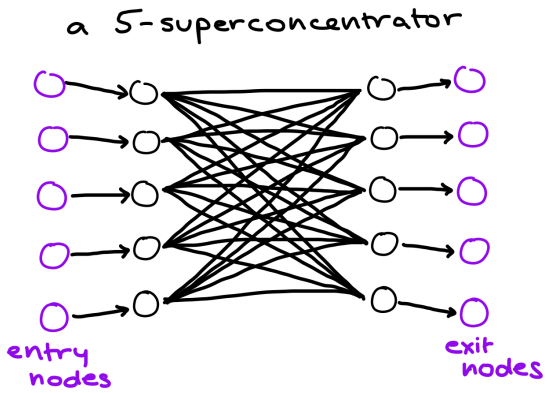
To prove this, we will take quite a long and casual stroll, stopping at various points along the way to smell the roses. For the entire first part of this paper, we will explore properties of a special type of graph, called

a **superconcentrator**, showing that we can construct superconcentrators without using too many edges. Then, we will construct what we call a **super-duper concentrator** by stacking these superconcentrators in a specific way, and spend the rest of this paper proving that super-duper concentrators give us the lower bound we want on the pebble game.

2 Superconcentrators

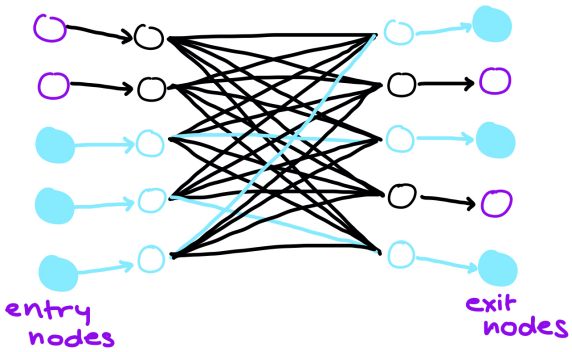
We will now construct a type of graph called a superconcentrator, which we can think of abstractly as graphs with many disjoint paths from the entry to the exit; these will be helpful for making a hard-to-pebble graph because an appropriately constructed graph would force us to place pebbles on each of these disjoint paths.

Definition 2.1. A **n-superconcentrator** is a directed acyclic graph, where some subset of n nodes are chosen to be the "entry" nodes, and some subset of n nodes are chosen to be the "exit" nodes, such that for any subset of $1 \leq k \leq n$ entry nodes and k exit nodes, we can find k node-disjoint paths mapping each entry node in the subset to some exit node in the subset.



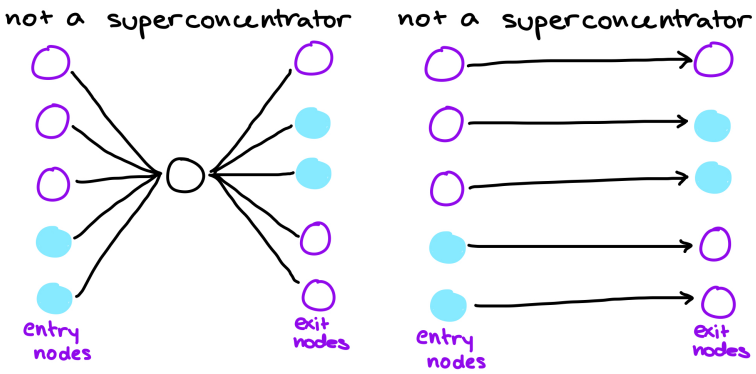
(assuming every edge in the picture is a directed edge going from left to right, the above is a 5-superconcentrator)

Another way of phrasing this condition is that in our superconcentrator, we can take any subset A of the entry nodes, and any equal-sized subset B of the exit nodes, and there is some one-to-one mapping $f : A \to B$ such that for every node $a \in A$, there is some path $a \to f(a)$ that does not intersect with any of the other paths $a' \to f(a')$.

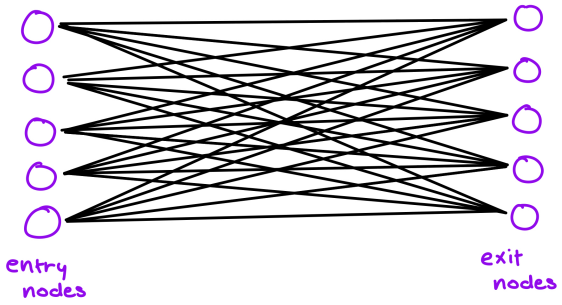


(the above is an example of two subsets of the exit and entry nodes and our node-disjoint paths connecting the nodes)

Example 2.2. The graph on the left is not a superconcentrator because no paths we can find connecting the blue nodes will be *node-disjoint*; they will all go through the center node. The graph on the right is not a superconcentrator because there are no two paths connecting the blue subsets.



Clearly, with more edges, it is easier to construct a superconcentrator; for example, with n^2 edges we can construct the superconcentrator where every entry node has an edge to every exit node.

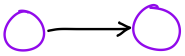


Our goal is to construct a superconcentrator with as few edges as possible.

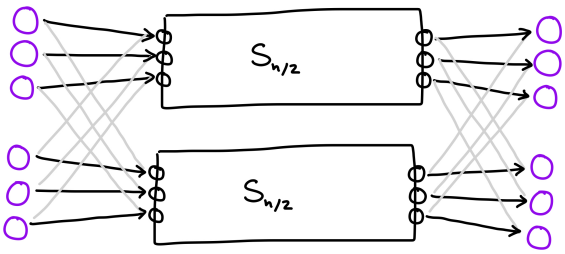
Remark 2.3. To be careful about notation, we will note here that n here is not the total number of vertices in our graph, but instead the number of entry and exit nodes in our graph; therefore, an n-superconcentrator always has at least $2n$ vertices, and the ones we are looking at will often have more. In the case we do need to denote the total number of vertices, we will use the variable v.

Example 2.4. Just to get our minds working in the correct direction, and because it is kind of a fun proof, we will first prove that for any n, we can construct a n-superconcentrator with $O(n \log n)$ edges.

Proof. We can construct such a superconcentrator using recursion. Specifically, we can construct a 1-superconcentrator using one edge, via the graph:



Then, if we say that S_k is the k-superconcentrator that we construct, we can construct our n-superconcentrator as:

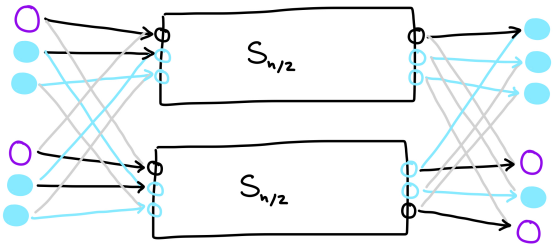


(where the grey edges are the same as the black edges, just colored differently so that the entire graph is easier to see)

First, we can see that if $g(k)$ is the number of edges in our k-superconcentrator, then $g(n) = 4n + 2g(n/2)$, since we have two $n/2$-superconcentrators, and then we add two edges for each of our entry nodes and two edges for each of our exit nodes. Moreover, $g(1) = 1$, so we can solve this recurrence relation to get that $g(n) = 4n \log n + n$, so $g(n) = O(n \log n)$, as we wanted.

Then, we need to prove that S_k is a superconcentrator for all k. We can again prove this by induction; at a base-case, S_1 is clearly a 1-superconcentrator.

For the inductive step, we look at S_n, and consider an arbitrary subset of k entry nodes and k exit nodes. We can see that we can split each subset into two, so that there are $j < k$ entry nodes mapping to the top $S_{n/2}$ and $k - j$ entry nodes mapping to the bottom $S_{n/2}$; similarly, we can connect j of the exit nodes of the top $S_{n/2}$ to j of the exit nodes in our subset and $k-j$ of the exit nodes of the top $S_{n/2}$ mapping to the remaining $k - j$ exit nodes in our subset. We can do this in such a way that no two nodes in our subset are connected to the same entry or exit node of their corresponding $S_{n/2}$, so that this start of our path is node-disjoint so far.



(an example mapping step for a given subset of the entry and exit nodes)

Then, by our inductive assumption $S_{n/2}$ is a superconcentrator, so we can find j node-disjoint paths connecting our selected nodes in the top $S_{n/2}$ and $k - j$ node-disjoint paths connecting our selected nodes in the bottom $S_{n/2}$. But this gives us our k node-disjoint paths connecting our entry and exit subsets, so we have shown that S_n is a superconcentrator.

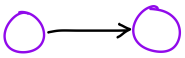
Thus, by induction, S_k is a superconcentrator for all k. □

Note that the superconcentrator we designed above is also a **permutation network**; it is stronger than a vanilla superconcentrator because for any subset of the entry and exit nodes, we can find k node-disjoint paths mapping the entry nodes to the exit nodes *in any order we want.*

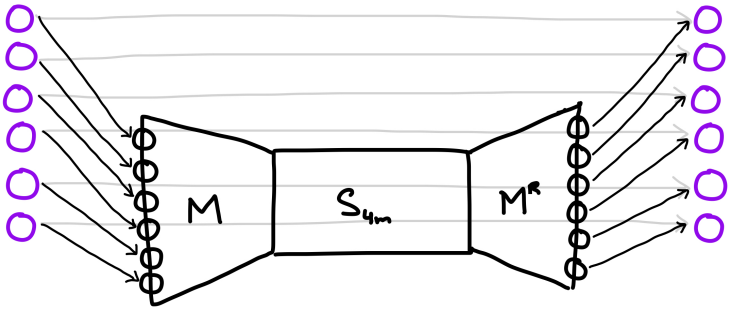
Perhaps this is a sign that the superconcentrator is more than what we asked for, and we can make a superconcentrator that's just the "bare minimum" by using fewer edges.

Theorem 2.5. There is a constant c such that for every n, there is a superconcentrator with cn edges.

We will build this superconcentrator recursively, as in the previous proof. Our 1-superconcentrator will be the same graph as before:



Then, to construct our superconcentrator S_n, we will assume $n = 6m$ for some m, and use S_{4m} as a backbone. To turn S_{4m} into a $6m$-superconcentrator, we will use a graph M, with $6m$ inputs and $4m$ outputs, which is not itself a superconcentrator but has some desirable properties that will allow our entire construction to be a superconcentrator. We will also use the graph M^R, which is just M but reversed. Specifically, our construction looks like the following:



In order for this to be a superconcentrator with cn edges, we need an M with very specific properties. In the following lemma, we will describe how M is structured.

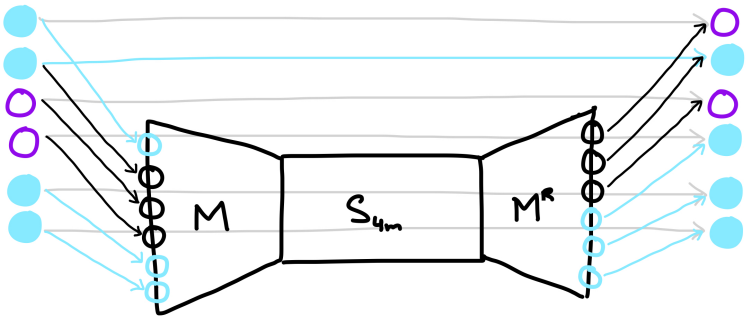
Lemma 2.6. There exists a constant c' such that for every $n = 6m$, we can construct a graph M with the following properties:

- M is a bipartite graph with $6m$ input nodes and $4m$ output nodes.
- For any subset of $k \leq 3m$ input nodes, there is some subset of k output nodes such that there are k node-disjoint paths from the input subset to the output subset.
  (Note that this is different from the definition of a superconcentrator, which requires us to have node-disjoint paths to all k-subsets of the output, rather than just one.)
- M has at most $c'n$ edges.

We will not prove this lemma in this paper; the proof is long and somewhat tangential to our ultimate goal of learning more about the pebble game. The idea behind the proof of this lemma is probabilistic. We look at all bipartite graphs M with $6m$ input nodes and $4m$ output nodes, where every output node has in-degree 9 and every input node has out-degree 6. Then, using a property called the Marriage Theorem, we can count the number of such graphs that do not fulfill the properties of Lemma 2.6. We find that the probability that a random bipartite graph constructed in this way fulfills Lemma 2.6 is greater than zero, which means there must exist some graph M that we want.

Now that we have this lemma, we can finish our proof of Theorem 2.5. Specifically, we can first show that this graph is a superconcentrator for every n. We can show this inductively. At a base case, S_1 is clearly a superconcentrator.

For the inductive case, we are trying to show that S_n is a superconcentrator. We consider arbitrary subsets A of k entry nodes and B of k exit nodes. Then, note that if $k > 3m$, we know that by the Pigeonhole Principle, because A and B both contain more than $n/2$ nodes, there must be at least one node in A that has a horizontal (grey) edge connecting it to a node in B. Thus, while $k > 3m$, we can pair off at least $k - 3m$ of the nodes simply by using the grey horizontal edges; these are clearly node-disjoint paths.



Thus, we have reduced to the case of finding node-disjoint paths connecting subsets A' and B' of the entry and exit nodes, respectively, where $|A'| = |B'| \leq 3m$. We know by Lemma 2.6 that there is some subset of output nodes of M such that there are k' node-disjoint paths from A' to these output nodes. By reversing this, we can also see that there are there are k' node-disjoint paths from some input nodes of M^R to B'. What's left is to map these output nodes to the input nodes. But this is just the problem of finding k' node-disjoint paths along S_{4m}, which we know we can do because S_{4m} is a superconcentrator, by the inductive assumption. Thus, by induction, S_n is an n-superconcentrator for every n. □

We will also show by induction that these superconcentrators have at most cn edges, where $c = 6c' + 9$ and $c'n$ is the number of edges in our subgraph M. At a base case, we can see that S_1 has 1 edge. For the inductive case, we can see that the number of edges in any S_n is $3n$, plus the number of edges in M and M^R, plus the number of edges in S_{4m}. By the inductive assumption, S_{4m} has $4c'n + 6n$ edges, and by Lemma 2.6, M and M^R each have $c'n$ edges. So, in total, our S_n has

$$3n + 4c'n + 5n + 2c'n = (6c' + 9)n = cn$$

edges. Thus, by induction, the number of edges in our graph is linear in n. □

We have now constructed a family of superconcentrators that have linearly many edges!

But our pebble game upper bound only applies to graphs of **constant in-degree**; that is, we care about families of graphs where we can find a constant c such that every vertex in all of our graphs has in-degree at most c. In order to use these superconcentrators to prove our pebble game lower bounds, it will also be helpful to verify that they have constant in-degree, and to count the number of vertices they have.

Proposition 2.7. The superconcentrators constructed in Theorem 2.5 are all graphs with in-degree at most 9.

Proof. We will again prove this by induction. Our superconcentrator S_1 has in-degree 1. Then, for our inductive case, we can look at each set of nodes in S_n separately. The exit nodes clearly have in-degree 2. Then, by our construction in Lemma 2.6, every output node in M^R has in-degree 6. The input nodes for M^R are the output nodes in S_{4m}, and by our inductive assumption, we know that every node besides the entry nodes for S_{4m} have in-degree at most 9. The entry nodes for S_{4m} are the output nodes for M, which by our construction have in-degree 9. Finally, the input nodes for M have in-degree 1. Thus, in total, the nodes in S_n have in-degree at most 9, so by induction our superconcentrators all have in-degree at most 9. □

Proposition 2.8. The number of vertices in the n-superconcentrator we construct is $O(n)$.

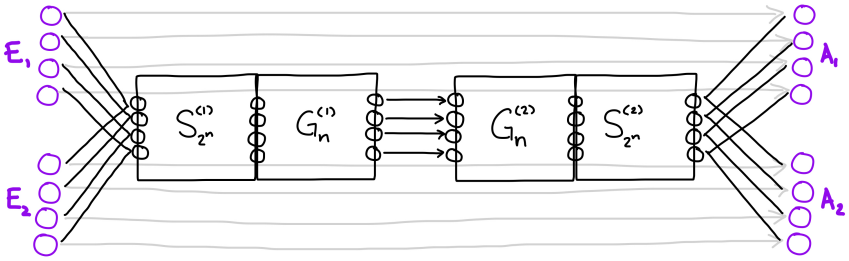
Proof. Note that every vertex has at least one edge. Thus, since each edge connects exactly two vertices, the number of vertices is at most twice the number of edges. By Theorem 2.5 there are cn edges in our superconcentrator S_n; this implies there are at most $2cn = O(n)$ vertices in each S_n. □

Now that we have our linear-sized superconcentrators, we have the building blocks to construct a hard-to-pebble graph.

3 Super-Duper Concentrators

Now that we have our linear superconcentrators, we can stack them to create our super-duper concentrators, which we will prove are hard to pebble. We will first describe how to construct these super-duper concentrators, and then once we have these graphs built, we will analyze various properties of these graphs and of the pebble game. Finally, we will build on this knowledge to prove our $v/\log v$ lower bound for the pebble game!

Definition 3.1. We define our super-duper concentrators recursively. At a base case, we start with G_8, which we define to be S_{2^8} from our previous section, or our linearly-sized 256-superconcentrator. Then, once we have each G_n, we can build G_{n+1} as:



Here, G_{n+1} has 2^{n+1} entry vertices and 2^{n+1} exit vertices, which are both divided into two sets (E_1 and E_2, and A_1 and A_2, respectively). Moreover, each entry has an edge directly to its corresponding exit, and to an entry node for $S^{(1)}_{2^n}$. The exit nodes for $S^{(1)}_{2^n}$ are the entry nodes for $G^{(1)}_n$, each exit node for $G^{(1)}_n$ has an edge directly to its corresponding entry node in $G^{(2)}_n$, and each exit node for $G^{(2)}_n$ is also an entry node for $S^{(2)}_{2^n}$. Finally, each exit node in $S^{(2)}_{2^n}$ has an edge directly to its corresponding vertex in A_1, and to its corresponding vertex in A_2.

Then, in order to prove pebble game results about G_n, we need to know that it is a constant in-degree graph and we need to know the number of vertices in our graph.

Proposition 3.2. The super-duper concentrators we constructed all have in-degree at most 9.

Proof. We can prove this inductively, using the fact that Proposition 2.7 tells us that all our linear superconcentrators have in-degree at most 9. At a base case, since $G_8 = S_{256}$, Proposition 2.7 gives us the result we want. Then, we can see that for the inductive case, in our graph G_n, our last layer A_1 and A_2 have in-degree 2. Then, by our inductive assumption and Proposition 2.7 our superconcentrator and G_{n-1} layers have in-degree at most 9, and finally the entry layer for $G^{(1)}_{n-1}$ clearly has in-degree 2. Thus, by induction, these constructed super-duper concentrators also have in-degree at most 9. □

Proposition 3.3. There is a constant c such that for all n, G_n has at most $cn2^n$ vertices.

Solution. Again we will prove this inductively. For a base case, we can see that since $G_8 = S_{256}$, Proposition 2.8 tells us that there exists some constant c' such that it has $c'2^n$ vertices. Then, for our inductive case, we can see that since G_n is composed of the entry vertices, the exit vertices, two copies of $S_{2^{n-1}}$, and two copies of G_n, by Proposition 2.8 and the inductive assumption, we get that the number of vertices in G_n is at most

$$2(2^n + c'2^{n-1} + c(n-1)2^{n-1}) = (c(n-1) + 2 + 2c')2^n,$$

so taking $c = 2 + 2c'$, we get the desired result.
Then, we have a c such that for every n, G_n has at most $cn2^n$ vertices. This means that we want to show it takes asymptotically $(cn2^n)/\log(cn2^n)$ pebbles, or that there exists some constant c' such it requires $c'2^n$ pebbles to win the pebble game on G_n.

But we've been sort of unclear about what it means to win the pebble game on G_n. Specifically, the way we required the pebble game at the beginning required a specific sink node that we were trying to pebble. From now on, we will try to accomplish a goal of pebbling a *set* of vertices, by which we mean find a play of the pebble game such that for each vertex in our set, there is some point in time during which we have a pebble on that vertex. So, we don't need to have pebbles on our entire set at once - it is fine to pebble one or a few at a time.

Definition 3.4. Here is some slightly confusing notation: we will say that a vertex **has been pebbled** if there was a pebble on that vertex at some point in the past. We distinguish this from the situation when a vertex currently **has a pebble**.

In order to translate between our original version of the game and the conditions under which our proof works, we note that

Proposition 3.5. If it takes a minimum of k pebbles to pebble a set A, then there must be some $a \in A$ such that it takes at least k pebbles to win our pebble game with a as the sink.

Proof. This is quite a simple proof by contradiction; if for every $a \in A$ we could win the pebble game with strictly less than k pebbles, then we could pebble A with less than k pebbles simply by pebbling each vertex one at a time. □

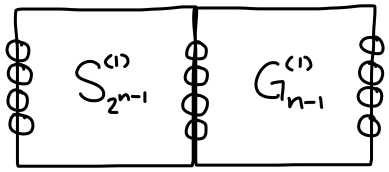
Thus, to prove our $v/\log v$ lower bound, it is enough to show that there is some constant c' such that for every G_n, there is some subset of the vertices that take $c'2^n$ pebbles to pebble.

To do so, we will use the fact that many subsets of G_n are superconcentrators. A quick exercise to think about is:

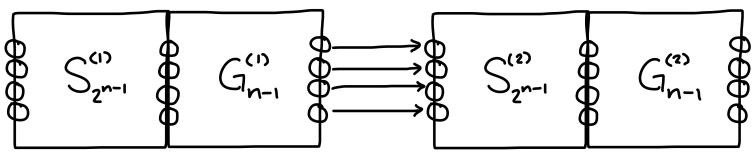
Exercise 3.6. Show that for $n > 8$, G_n is *not* a superconcentrator. That is, find some subset of k source nodes of G_n and some subset of k sink nodes of G_n such that we cannot find k node-disjoint paths between the source and sink nodes.

However, there are many superconcentrator subsets of G_n. For all of these, the proof that they are superconcentrators follows directly from the fact that each S_n is a superconcentrator, but since there are so many such subsets, we will just show two examples, and leave the rest as an exercise.

Example 3.7. For any G_n, the following subgraph is a superconcentrator:



For any subset of k entry vertices and k exit vertices, we can see that there are k exit vertices of $S^{(1)}_{2^{n-1}}$ that have edges directly to the k exit vertices, by the structure of $G^{(1)}_{n-1}$. Then, there are k node-disjoint paths from the entry vertices to the k exit vertices of $S^{(1)}_{2^{n-1}}$ by the fact that this part is a superconcentrator, so we are done.
Then, the following subgraph is also a superconcentrator:



We just showed that the first part is a superconcentrator, and since the second part is the same thing but reversed, it is also a superconcentrator. Then, for any set A of k entry vertices and B k exit vertices, we can pick an arbitrary set C of k exit vertices of $G^{(1)}_{n-1}$. Then, we know that since the first part is a superconcentrator, there are k node-disjoint paths from A to C. Then, there are direct edges from C to a corresponding set C' of entry vertices of $G^{(2)}_{n-1}$, and since the second part is a superconcentrator, there are k node-disjoint paths from C' to B, and we are done.

Exercise 3.8. List all the subgraphs of G_n, built out of the blocks E_1, E_2, $S^{(1)}_{2^{n-1}}$, $G^{(1)}_{n-1}$, $G^{(2)}_{n-1}$, $S^{(2)}_{2^{n-1}}$, A_1, and A_2, which are superconcentrators.

For the rest of this section, we will claim without proof that various subgraphs of G_n are superconcentrators, as they become helpful to us.

Ok, but why is this helpful? We said before that we like superconcentrators because it seems like the number of distinct nodes we need in order to win the pebble game should be decently high, since we can find so many node-disjoint paths. We will formalize this idea now.

Lemma 3.9. If we have an n superconcentrator with $j < n$ pebbles on some j vertices, and A is a subset of at least $j + 1$ exit nodes, then there are at least $n - j$ entry nodes with pebble-free paths to A.

Proof. We can prove this by contradiction. Assume there aren't at least $n - j$ entry nodes with pebble-free paths to A; that is, we can find $j + 1$ entry nodes such that none of them have a pebble free path to A. But since we are in a superconcentrator, we know that we can find $j + 1$ node-disjoint paths from these entry nodes to A. Since there are only j pebbles, we know at least one of these paths must be pebble-free, which contradicts our assumption. Thus, there are at most j entry nodes with *no* pebble-free paths to A, which means there are at least $n - j$ entry nodes with pebble-free paths to A. □

This lemma will be crucial in proving our final result, which we are now ready to approach.

Theorem 3.10 (Paul-Tarjan-Celoni [PTC76])**.** For every $n \geq 8$, we can find a sink node such that it takes $c'2^n$ pebbles to win the pebble game on G_n, where $c' = 2^{-8}$.

As in most of the proofs in this paper, we will prove this by induction. But we will need a much stronger inductive assumption to be able to do so, so instead we will prove the following statement, defining α_n to be 2^{n-8}:

Theorem 3.11. For every graph G_n, in order to pebble any set A of at least $14\alpha_n$ exit nodes, if we start with pebbles on at most $3\alpha_n$ nodes, then we can find an interval of time during which we pebble $34\alpha_n$ entry nodes and, for the entire interval, we have at least α_n nodes on the graph.

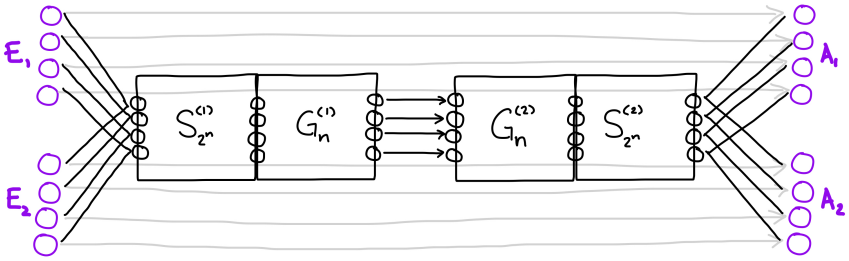
That is, we look at every possible strategy that starts with these $3\alpha_n$ pebbles and pebbles all $14\alpha_n$ sink nodes. We will consider each strategy as a series of timesteps $[0, t]$, where at each timestep we either place a pebble on a node or remove a pebble from a node. Then, we will show that we can find some time interval $[t_1, t_2] \subseteq [0, t]$ such that for the entire time interval, we have at least α_n nodes on the graph, and during the time interval, we pebble some $34\alpha_n$ entry nodes (in any order, and not necessarily at the same time).

Proof. The rest of this paper will be dedicated to proving this theorem.

We start with the base case of when $n = 8$. In that case, we are working directly with a superconcencentrator, and we want to show that if we start with 3 pebbles and want to pebble some set A of 14 exit nodes, then there must be an interval of time during which we pebble at least 34 entry nodes and there is at least

one pebble on the graph for the entirety of this interval. But then, consider some subset A' that consists of 4 exit nodes from A. Since G_8 is a 256-superconcentrator, we can apply Lemma 3.9 with $j = 3$ to see that there must be at least 253 entry nodes with pebble-free paths to A'. But by the pigeonhole principle, that means there must be some $a \in A'$ with at least $64 > 34$ entry nodes with pebble-free paths to a. We know that in order to pebble a, we must first pebble these 34 entry nodes. So if we let t_2 be the time at which we pebble a and t_1 be the *last* time we start pebbling these 34 entry nodes before that, then we can see that $[t_1, t_2]$ must be an interval of time during which we pebble at least 34 entry nodes and always have at least one pebble on the graph (If we take all pebbles off the graph, we have essentially reset the state of the game, and must pebble all the entry nodes again to pebble a - this contradicts the fact that we chose t_1 to be the last time we pebble the entry nodes before we pebble a.) Thus, Theorem 3.11 holds for the base case.

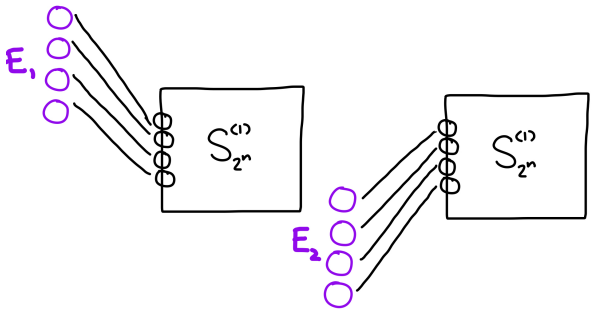
The inductive case is slightly more complicated, and we end up having to do a bit of casework. We will show that each strategy to pebble a graph G_{n+1} falls into one of four options, and then show that for each of the options, our inductive hypothesis holds.



As a reminder, we want to show that if we want to pebble a set A of at least $14\alpha_{n+1} = 28\alpha_n$ nodes, and we start with pebbles on $3a_{n+1} = 6a_n$ nodes, then we can find an interval of time during which we pebble $34\alpha_{n+1} = 68\alpha_n$ entry nodes and have $2\alpha_n$ nodes on the graph the entire time.

Case 1: There is a time interval $[t_1, t_2]$ such that for the entire interval, there are at least $3\alpha_n$ pebbles on G_{n+1} and during the interval, we pebble at least $7\alpha_n$ entry nodes of $G_n^{(1)}$.

Let A be the set of these $7\alpha_n$ entry nodes of $G_n^{(1)}$. To prove the inductive hypothesis for this case, we note that A is also a set of exit nodes of $S_{2^n}^{(1)}$. We want to apply Lemma 3.9 to the following two superconcentrators:



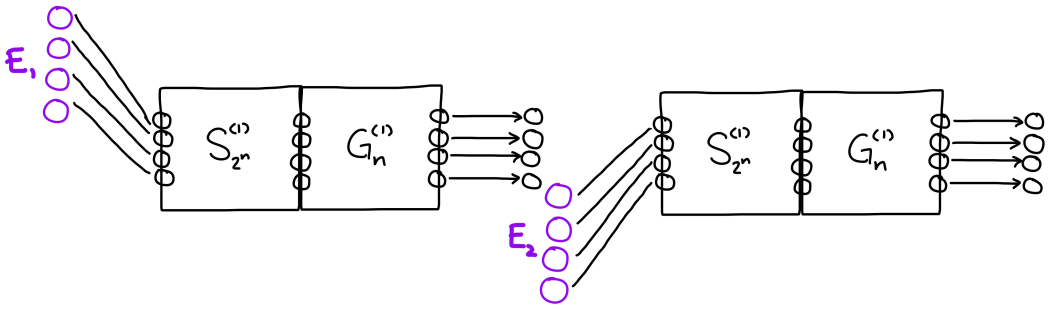
But in order to apply this lemma, we need an upper bound on how many pebbles are on this graph to start out with. To get this upper bound, we simply extend our interval a bit at the beginning. Specifically, let t_0 be the *last* time before t_1 during which there at most $6\alpha_n$ pebbles on the graph. That is, if $t_0 < t_1 - 1$, then in the interval $[t_0 + 1, t_1 - 1]$, there are always at least $6\alpha_n + 1$ pebbles on the graph. We can see that whether or not $t_0 < t_1 - 1$, there are always at least $3\alpha_n$ pebbles at time $t_0 + 1$, so there must be at least $3\alpha_n - 1$ pebbles at time t_0. Thus, in the interval $[t_0, t_2]$ there are always at least $3\alpha_n$ pebbles on the graph, and we start with at most $6\alpha_n$ pebbles.

Now, we can apply Lemma 3.9 to the above two superconcentrators, with $j = 6\alpha_n$. This tells us that there are $2^n - 6\alpha_n = 250\alpha_n$ nodes in E_1, and another $250\alpha_n$ nodes in E_2, with pebble-free paths to A at time t_0. Because all these nodes must get pebbled before A can get pebbled, we have that these $500\alpha_n$ nodes must get pebbled during the time interval $[t_0, t_2]$.

Thus, we have found a time interval during which there are $3\alpha_n - 1 > \alpha_{n+1}$ pebbles on the graph the entire time, and during which $500\alpha_n > 34\alpha_{n+1}$ entry nodes of G_{n+1} get pebbled. So our theorem holds for this case.

Case 2: There is a time interval $[t_1, t_2]$ such that for the entire interval, there are at least $3\alpha_n$ pebbles on G_{n+1} and during the interval, we pebble at least $7\alpha_n$ entry nodes of $G_n^{(2)}$.

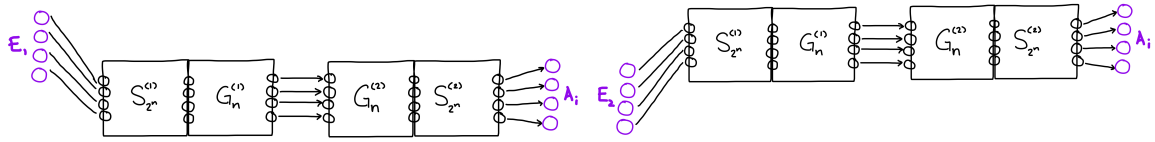
This is essentially the same as Case 1, except we are working with the entry nodes of $G_n^{(2)}$ instead of the entry nodes of $G_n^{(1)}$. That is, we again consider the interval $[t_0, t_2]$, where t_0 is the last time before t_1 when there are at most $6\alpha_n$ pebbles on the graph. Then, we apply Lemma 3.9 to the following two superconcentrators:



This lemma tells us that there are $250\alpha_n$ nodes of E_1 and $250\alpha_n$ nodes of E_2 that get pebbled during $[t_0, t_2]$, and since, as before, there are always at least $3\alpha_n - 1$ pebbles on the graph during this interval, we have found a time interval that satisfies our theorem for this case.

Case 3: There is a time interval $[t_1, t_2]$ such that for the entire interval, there are at least $3\alpha_n$ pebbles on G_{n+1} and during the interval, we pebble at least $14\alpha_n$ exit nodes of G_{n+1}.

This is again very similar to the previous two cases. First, we note that by the pigeonhole principle, there is some i such that we are pebbling at least $7\alpha_n$ nodes of A_i. Then, we pick at time t_0 as before, so that there at most $6\alpha_n$ pebbles on the graph at time t_0 and during $[t_0, t_2]$ there are always at least $3\alpha_n - 1$ pebbles on the graph during this time interval. Moreover, applying Lemma 3.9 to the following two superconcentrators

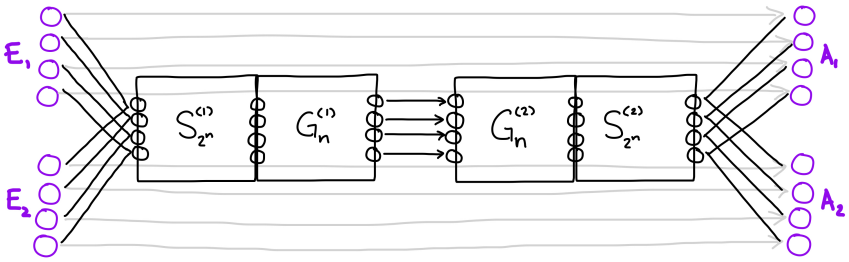


tells us that we pebble $500\alpha_n$ entry nodes of G_{n+1} during this time, so our theorem also holds for this case.

Case 4: None of the first three cases hold.

(You may have noticed that we haven't actually used our inductive hypothesis yet - this is the only case that requires the inductive hypothesis.)

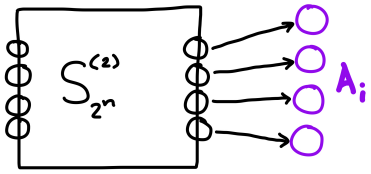
As a reminder, we are trying to pebble some subset of the exit nodes on the following graph:



Let $[0, t]$ be the time interval during which we start with pebbles on at most $3\alpha_{n+1}$ nodes and end having pebbled $14\alpha_{n+1}$ exit nodes.

Then, let $[0, t_1]$ be the interval during which the first $14\alpha_n$ exit nodes are pebbled. Since we are not in **Case 3**, there must be some time $t_2 \in [0, t_1]$ such that there are less than $3\alpha_n$ pebbles on the graph.

Then, we note that since there are $14\alpha_n$ exit nodes left to be pebbled after t_1, there must be some A_i such that we pebble $7\alpha_n$ nodes in A_i in the interval $[t_1, t]$, by the pigeonhole principle. Applying Lemma 3.9 to the following superconcentrator, with $j = 3\alpha_n$:



we get that there are $2^n - 3\alpha_n = 253\alpha_n$ entry nodes of $S^{(2)}_{2^n}$ with pebble-free paths to these $7\alpha_n$ exit nodes; each of these must be pebbled in the time interval $[t_1, t]$.

Then, we note that in $[t_1, t]$ we pebble $253\alpha_n$ exit nodes of $G^{(2)}_n$ and we start with less than $3\alpha_n$ pebbles on the graph. Thus, we can apply the **inductive assumption** to $G^{(2)}_n$ to see that there must be some time interval $[t_2, t_3] \subseteq [t_1, t]$ such that there are always at least α_n pebbles on $G^{(2)}_n$ and at least $34\alpha_n$ entry nodes of $G^{(2)}_n$ are pebbled.

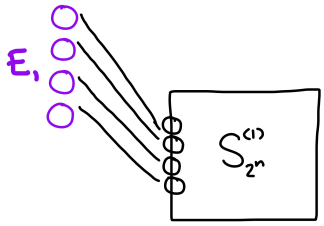
Let $[t_2, t_4]$ be the time interval during which the first $7\alpha_n$ of these entry nodes are pebbled. Since we are not in **Case 2**, we can see that there must be some $t_5 \in [t_2, t_4]$ such that there are less than $3\alpha_n$ pebbles on the graph at time t_5.

Then, during $[t_5, t_3]$, the remaining $27\alpha_n$ entry nodes of of $G^{(2)}_n$ are pebbled, so during $[t_5, t_3]$ we pebble $27\alpha_n$ exit nodes of $G^{(1)}_n$ and we start with less than $3\alpha_n$ pebbles on the graph. Thus, the **inductive assumption** tells us there is some time interval $[t_6, t_7]$ such that there are always at least α_n pebbles on $G^{(1)}_n$ and at least $34\alpha_n$ entry nodes of $G^{(1)}_n$.

Note that since there are α_n pebbles on $G^{(1)}_n$ and α_n pebbles on $G^{(2)}_n$, we are now within a time interval where there are always at least α_{n+1} pebbles on the graph.

Then, using an analogous argument from before, since we are not in **Case 1**, there must be a time $t_8 \in [t_6, t_7]$ such that at time t_8 there are less than $3\alpha_n$ pebbles on the graph, and within the interval $[t_8, t_7]$, we must pebble $27\alpha_n$ entry nodes of $G^{(1)}_n$.

Then, applying Lemma 3.9 to the following superconcentrator:



with $j = 3\alpha_n$ tells us that there are at least $2^n - 3\alpha_n = 253\alpha_n$ vertices in E_1 that have pebble-free paths to these $27\alpha_n$ exit nodes of $S_{2^n}^{(1)}$.

These must all get pebbled during this interval in order for the exit nodes to get pebbled, so we have found a time interval where there are always at least α_{n+1} nodes with pebbles on them, and during which we pebble $253\alpha_n > 34\alpha_{n+1}$ entry nodes, so our inductive hypothesis holds for this case as well.

Thus, in any of the four cases, our inductive hypothesis holds, and by induction we can say this theorem is true for all our graphs G_n. □

Moreover, by taking Theorem 3.11 in combination with Proposition 3.5 shows that Theorem 3.10 holds, so we have proved our $v/\log v$ lower bound!

4 Conclusion

The pebble game is a weird and important way for us to learn about lower bounds for space complexity of algorithms, especially in the tradeoff between space and time complexity. However, it is also interesting in its own right, because exploring properties of the pebble game allows us to further develop and understand strange and interesting graphs. I hope you found our little journey to build one such strange and interesting graph as fun as I did!

References

[HPV77] John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *J. ACM*, 24(2):332–337, apr 1977.

[PTC76] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, page 149–160, New York, NY, USA, 1976. Association for Computing Machinery.

[SP98] Uwe Schoning and Randall Pruim. *Gems of Theoretical Computer Science*. Springer-Verlag, 1998.