Математическое сочинение:

“Индекс Винера в растущих графах”

Чудопалов Денис
ВМК МГУ

Февраль 2026

Содержание

1 Введение

В данной работе рассматривается способы эффективно делать расчёт индекса Винера для растущего графа, в случае присоединения к нему узла или рёбра. Это может быть использванно для многих практических целей и задач имеющих динамическую природу: компьютерные сети, химические соединения, топологией молекул белка.

1.1 Постановка задачи

Дано:

Неориентированный граф $G(V, E)$, где V - множество вершин, E - множество ребер.

Задача:

- Дать определение индексу Винера
- Найти аналитические решения
- Разобрать удобные частные случаи
- Написать программу для вычисления индекса Винера

2 Индекс Винера

На протяжении всей работы все графы будут неориентированными

Определение 1. Пусть G — граф с множеством вершин $V(G)$ и множеством рёбер $E(G)$. Расстояние $d_G(u,v)$ между двумя вершинами $u, v \in V(G)$ — это минимальное число рёбер на пути в G между u и v.

Определение 2. Пусть G удовлетворяет определению 1. *Винеровский индекс* $W(G)$ графа G определяется формулой

$$W(G) = \sum_{\{u,v\} \subseteq V(G)} d_G(u,v),$$

а *среднее расстояние* $\mu(G)$ между вершинами графа G — формулой

$$\mu(G) = \frac{W(G)}{\binom{|V(G)|}{2}}.$$

Определение 3. Пусть G удовлетворяет определению 1. *Расстояние вершины* v обозначим через $d_G(v)$ и определим как сумму расстояний между v и всеми остальными вершинами графа G:

$$d_G(v) = \sum_{u \in V(G) \setminus \{v\}} d_G(u,v).$$

Таким образом, индекс Винера можно определить немного иначе:

$$W(G) = \frac{1}{2} \sum_{v \in V(G)} d_G(v),$$

где множитель $\frac{1}{2}$ компенсирует тот факт, что каждый путь между u и v учитывается в $d_G(u)$, а также в $d_G(v)$.

Определение 4. Пусть G — граф. *Диаметр* $d(G)$ графа G определяется как

$$d(G) = \min_{\{u,v\} \subseteq V(G)} d_G(u,v).$$

2.1 Пример вычисления индекса Винера

Рассмотрим граф T на 10 вершинах (рис. 1):

$$E(T) = \{(v_1,v_2),(v_2,v_3),(v_3,v_4),(v_4,v_5),(v_5,v_6),(v_3,v_7),(v_3,v_8),(v_5,v_9),(v_5,v_{10})\}.$$

Индекс Винера считаем по формуле

$$W(T) = \frac{1}{2} \sum_{v \in V(T)} d_T(v), \qquad d_T(v) = \sum_{u \neq v} d_T(u,v).$$

Для данного графа получаем:

$$d_T(v_1) = 31, \quad d_T(v_2) = 23, \quad d_T(v_3) = 17, \quad d_T(v_4) = 19, \quad d_T(v_5) = 19,$$
$$d_T(v_6) = 27, \quad d_T(v_7) = 25, \quad d_T(v_8) = 25, \quad d_T(v_9) = 27, \quad d_T(v_{10}) = 27.$$

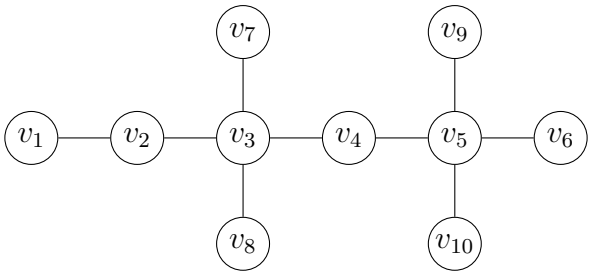


Рис. 1: граф T на 10 вершинах.

Тогда

$$\sum_{i=1}^{10} d_T(v_i) = 240 \quad \Rightarrow \quad W(T) = \frac{1}{2} \cdot 240 = 120.$$

Среднее расстояние между вершинами:

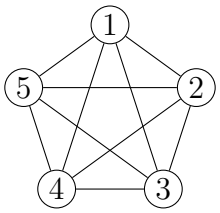
$$\mu(T) = \frac{W(T)}{\binom{10}{2}} = \frac{120}{45} = \frac{8}{3}.$$

3 Аналитические решения

Первое что приходит в голову после постановки задачи для растущих графов, возможно ли вычислять Индекс Винера с нимиальными затратами? На этот вопрос есть ответ - если индекс можно выразить аналитически, те существует конечная формула. Далее будем рассматривать такие графы.

3.1 Полный граф K_n

Рисунок (пример K_5)



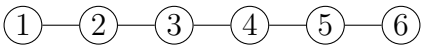
Вывод формулы

В полном графе K_n любая пара различных вершин соединена ребром, значит $\mathrm{dist}(x, y) = 1$ для всех $x \neq y$. Число неупорядоченных пар равно $\binom{n}{2}$, поэтому

$$\boxed{\mathrm{W}(K_n) = \binom{n}{2} = \frac{n(n-1)}{2}.}$$

3.2 Путь P_n

Рисунок (пример P_6)



Вывод формулы

Пронумеруем вершины $1, 2, \ldots, n$ вдоль пути. Тогда

$$\mathrm{dist}(i, j) = |i - j|.$$

Для каждого расстояния $d \in \{1, \ldots, n-1\}$ число пар вершин на расстоянии d равно $n - d$ (это пары $(i, i + d)$, где $i = 1, \ldots, n - d$). Тогда

$$\mathrm{W}(P_n) = \sum_{d=1}^{n-1} d(n-d) = n \sum_{d=1}^{n-1} d - \sum_{d=1}^{n-1} d^2.$$

Используем известные суммы:

$$\sum_{d=1}^{n-1} d = \frac{(n-1)n}{2}, \qquad \sum_{d=1}^{n-1} d^2 = \frac{(n-1)n(2n-1)}{6}.$$

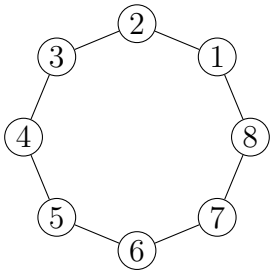
Подставляя, получаем

$$\mathrm{W}(P_n) = n \cdot \frac{(n-1)n}{2} - \frac{(n-1)n(2n-1)}{6} = \frac{(n-1)n(n+1)}{6} = \binom{n+1}{3}.$$

$$\boxed{\mathrm{W}(P_n) = \binom{n+1}{3} = \frac{(n-1)n(n+1)}{6}.}$$

3.3 Цикл C_n

Рисунок (пример C_8)



Вывод формулы

В цикле расстояние между двумя вершинами равно меньшей длине дуги между ними. Положим $k = \lfloor n/2 \rfloor$.

Случай 1: $n = 2k + 1$ **(нечётный).** Для каждого $d = 1, \ldots, k$ каждая вершина имеет ровно 2 вершины на расстоянии d, поэтому число неупорядоченных пар на расстоянии d равно n. Тогда

$$\mathrm{W}(C_n) = \sum_{d=1}^{k} d \cdot n = n \cdot \frac{k(k+1)}{2}.$$

Так как $n = 2k + 1$, то $k(k+1) = \frac{(n^2-1)}{4}$, и

$$\boxed{\mathrm{W}(C_{2k+1}) = \frac{(2k+1)\,k(k+1)}{2} = \frac{n(n^2-1)}{8}.}$$

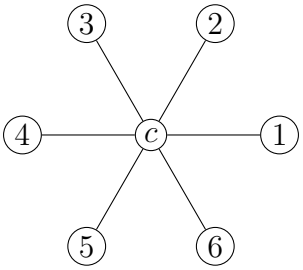
Случай 2: $n = 2k$ **(чётный).** Для $d = 1, \ldots, k-1$ число пар на расстоянии d равно n. Для $d = k$ (противоположные вершины) число пар равно $n/2$. Значит

$$\mathrm{W}(C_n) = \sum_{d=1}^{k-1} d \cdot n + k \cdot \frac{n}{2} = n \cdot \frac{(k-1)k}{2} + k \cdot k = k^3 = \frac{n^3}{8}.$$

$$\boxed{\mathrm{W}(C_{2k}) = k^3 = \frac{n^3}{8}.}$$

3.4 Звезда $S_n = K_{1,n-1}$

Рисунок (пример S_7)



Вывод формулы

В звезде есть центр c и $n-1$ листьев.

- Пар *центр–лист* ровно $n-1$, расстояние 1.
- Пар *лист–лист* ровно $\binom{n-1}{2}$, расстояние 2 (через центр).

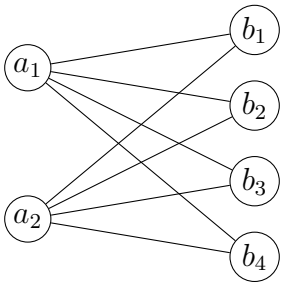
Следовательно,

$$\mathrm{W}(S_n) = (n-1)\cdot 1 + \binom{n-1}{2}\cdot 2 = (n-1) + (n-1)(n-2) = (n-1)^2.$$

$$\boxed{\mathrm{W}(S_n) = (n-1)^2.}$$

3.5 Полный двудольный граф $K_{a,b}$

Рисунок (пример $K_{2,4}$)



Вывод формулы

В $K_{a,b}$ любые две вершины из разных долей имеют расстояние 1, а любые две вершины из одной доли — расстояние 2.

- Междольных пар ab, вклад ab.
- Внутри доли размера a пар $\binom{a}{2}$, вклад $2\binom{a}{2}$.
- Внутри доли размера b пар $\binom{b}{2}$, вклад $2\binom{b}{2}$.

Итого

$$\boxed{\mathrm{W}(K_{a,b}) = ab + 2\left(\binom{a}{2} + \binom{b}{2}\right) = ab + a(a-1) + b(b-1).}$$

3.6 Полный многодольный граф $K_{n_1,\dots,n_k}$

Полный вывод формулы

В полном k-дольном графе:

- если вершины в разных долях, расстояние 1;
- если в одной доле, расстояние 2 (через любую вершину другой доли).

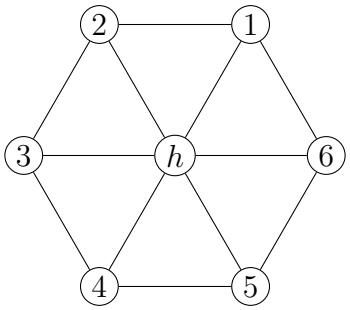
Пусть $n = \sum_{i=1}^{k} n_i$. Тогда число пар из разных долей равно $\sum_{i<j} n_i n_j$ (вклад 1), а число пар внутри долей равно $\sum_i \binom{n_i}{2}$ (вклад 2). Поэтому

$$\boxed{\mathrm{W}(K_{n_1,\dots,n_k}) = \sum_{1\le i<j\le k} n_i n_j \;+\; 2\sum_{i=1}^{k}\binom{n_i}{2}.}$$

Также можно переписать $\sum_{i<j} n_i n_j = \binom{n}{2} - \sum_i \binom{n_i}{2}$.

3.7 Колесо W_n (цикл на $n-1$ вершинах + центр)

Рисунок (пример W_7)



Вывод формулы

В колесе W_n:

- Центр h соединён со всеми $n-1$ вершин обода: вклад $n-1$ пар на расстоянии 1.
- На ободе есть $(n-1)$ соседних пар (рёбра цикла), расстояние 1, вклад $n-1$.
- Любые две *несоседние* вершины обода имеют расстояние 2 (через центр). Число всех пар на ободе: $\binom{n-1}{2}$, значит несоседних пар: $\binom{n-1}{2} - (n-1)$.

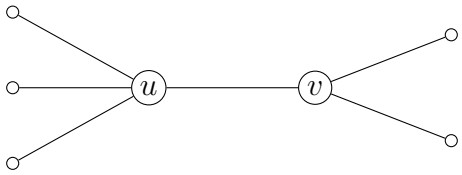
Итак,

$$\mathrm{W}(W_n) = (n-1) + (n-1) + 2\left(\binom{n-1}{2} - (n-1)\right) = 2\binom{n-1}{2} = (n-1)(n-2).$$

$$\boxed{\mathrm{W}(W_n) = (n-1)(n-2).}$$

3.8 Двойная звезда $DS_{a,b}$

Двойная звезда $DS_{a,b}$: две центральные вершины u и v соединены ребром, к u подвешено a листьев, к v — b листьев. Всего вершин $n = a + b + 2$.
Рисунок



Вывод формулы

Считаем вклад по типам пар:

- (u, v): одна пара на расстоянии 1.
- $(u, \text{лист у } u)$: a пар на расстоянии 1; $(v, \text{лист у } v)$: b пар на расстоянии 1.
- Два листа у u: $\binom{a}{2}$ пар на расстоянии 2; два листа у v: $\binom{b}{2}$ пар на расстоянии 2.
- Лист у u и вершина v: a пар на расстоянии 2; лист у v и вершина u: b пар на расстоянии 2.
- Лист у u и лист у v: ab пар на расстоянии 3.

Итого

$$\mathrm{W}(DS_{a,b}) = 1 + (a + b) + 2\left(\binom{a}{2} + \binom{b}{2}\right) + 2(a + b) + 3ab.$$

Упростим:

$$2\binom{a}{2} = a(a-1), \quad 2\binom{b}{2} = b(b-1),$$

поэтому

$$\boxed{\mathrm{W}(DS_{a,b}) = 1 + 3(a + b) + a(a-1) + b(b-1) + 3ab.}$$

3.9 Универсальный формула для графов диаметра 2

Пусть G — связный граф диаметра 2, то есть любые две вершины находятся на расстоянии 1 или 2. Пусть $n = |V(G)|$, $m = |E(G)|$. Тогда пар на расстоянии 1 ровно m (это рёбра), а остальных пар $\binom{n}{2} - m$, и они на расстоянии 2. Следовательно,

$$\mathrm{W}(G) = 1 \cdot m + 2\left(\binom{n}{2} - m\right) = 2\binom{n}{2} - m = n(n-1) - m.$$

$$\boxed{\text{Если } \operatorname{diam}(G) = 2,\ \mathrm{W}(G) = n(n-1) - m.}$$

Этот формула мгновенно даёт решения для многих классов (включая колёса, многие двудольные графы при $a, b \geq 2$ и т. п.).

3.10 Итог

Рассмотренные частные случаи делают вычисление индекса Винера элеметраной задачей, при каждом росте графа, нужно всего лишь подставлять значения и получать результат. Однако таких случаев не так много, поэтому далее рассмотрим чуть более общирный класс графов, который может представлять для нас интерес.

4 Случай деревьев

Особый интерес для нас представляют деревья. Поскольку в дереве путь, а значит и расстояние, между двумя вершинами единственен, индекс Винера для дерева вычислять существенно проще, чем для произвольного графа. Далее мы приведём различные формулы для вычисления индекса Винера: в первой части — прямые формулы, во второй — рекуррентные, которые требуют выполнения некоторых условий на деревья, но при их выполнении могут значительно упростить вычисления.

4.1 Прямые формулы для деревьев

Первая формула, которую мы приведём, является базовой и была получена Х. Винером в 1947 году. Если определение индекса Винера акцентирует внимание на сумме расстояний от каждой вершины до всех остальных, то данная формула подсчитывает, сколько раз в суммарных кратчайших путях используется каждое ребро.

Определение 5. Пусть $e = (u, v) \in E(T)$ — ребро дерева T. Поддеревья T_u и T_v определим как связные компоненты графа $T \backslash e$, содержащие соответственно вершины u и v. Числа вершин этих поддеревьев обозначим

$$n_u(e) = |V(T_u)|, \qquad n_v(e) = |V(T_v)|.$$

Теорема 1. *Пусть T — дерево. Тогда*

$$W(T) = \sum_{e=(u,v)\in E(T)} n_u(e)\, n_v(e). \tag{1}$$

Доказательство. Так как T — дерево, путь между любой вершиной $x \in V(T_u)$ и любой вершиной $y \in V(T_v)$ единственен и обязательно содержит ребро e. Если же вершины x и y выбираются из одной и той же компоненты, то ребро e в путь между ними не входит. Следовательно, величина $n_u(e)\, n_v(e)$ в точности равна числу пар вершин, кратчайший путь между которыми проходит через e. Суммируя $n_u(e)\, n_v(e)$ по всем рёбрам $e \in E(T)$, получаем индекс Винера $W(T)$. □

Определение 6. Пусть $e = (u, v) \in E(G)$ — ребро графа G. Определим множества вершин

$$B_u(e) = \{x \in V(G) \colon d_G(x, u) < d_G(x, v)\}, \qquad B_v(e) = \{y \in V(G) \colon d_G(y, v) < d_G(y, u)\}.$$

Мощности этих множеств обозначим

$$n_u(e) = |B_u(e)|, \qquad n_v(e) = |B_v(e)|.$$

Пример 1. Рассмотрим дерево T на рис. 2. Поскольку дерево не содержит циклов, каждое ребро является мостом: при удалении ребра e дерево распадается на две компоненты размерностей a и $10 - a$, и вклад ребра в индекс Винера равен $a(10 - a)$ (это частный случай формулы $W(T) = \sum_e n_u(e) n_v(e)$).

Посчитаем размеры компонент для каждого ребра (достаточно указать меньшую из двух компонент):

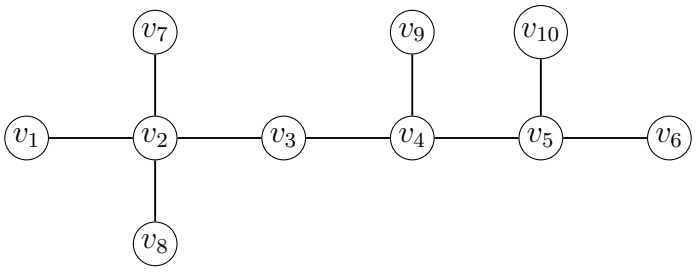


Рис. 2: Дерево T порядка 10.

ребро e	a	$a(10-a)$
(v_1, v_2)	1	$1 \cdot 9 = 9$
(v_2, v_3)	4	$4 \cdot 6 = 24$
(v_3, v_4)	5	$5 \cdot 5 = 25$
(v_4, v_5)	7	$7 \cdot 3 = 21$
(v_5, v_6)	1	$1 \cdot 9 = 9$
(v_2, v_7)	1	$1 \cdot 9 = 9$
(v_2, v_8)	1	$1 \cdot 9 = 9$
(v_4, v_9)	1	$1 \cdot 9 = 9$
(v_5, v_{10})	1	$1 \cdot 9 = 9$

Суммируя вклады, получаем

$$W(T) = 9 + 24 + 25 + 21 + 9 + 9 + 9 + 9 + 9 = 124.$$

4.2 Итерационные формулы для деревьев

В этом случае мы будем рассмотривать специальные поддеревья, чтобы получвть индекс Винера всего дерева как комбинацию индексов этих поддеревьев. Например, когда дерево T получается соединением нескольких копий дерева T' в одной вершине u, причём каждый раз используется одна и та же вершина $v' \in V(T')$. Далее будут приведены некоторые рекуррентные формулы.

Первая идея, которая возникает при желании вычислять индекс Винера рекурсивно — взять дерево, удалить лист и вычислить индекс Винера оставшегося поддерева.

Теорема 2. *Пусть T — дерево на $n \geq 2$ вершинах, а $v \in V(T)$ — лист дерева T. Пусть также $(u, v) \in E(T)$ и $T' = T - v$ — подграф дерева T, полученный удалением вершины v (и инцидентного ей ребра). Тогда*

$$W(T) = W(T') + d_{T'}(u) + n - 1.$$

Доказательство. Рассмотрим две вершины x и y дерева T. Если $v \neq x$ и $v \neq y$, то расстояние между x и y не изменяется после удаления v, поэтому сумма всех таких расстояний равна $W(T')$.

Если одна из вершин равна v, без ограничения общности пусть $x = v$. Тогда для любой вершины $y \in V(T')$ имеем

$$d_T(x, y) = d_T(v, y) = d_{T'}(u, y) + 1.$$

Суммируя по всем $y \in V(T')$ (таких вершин $n-1$), получаем вклад

$$\sum_{y \in V(T')} d_T(v,y) = \sum_{y \in V(T')} \bigl(d_{T'}(u,y) + 1\bigr) = d_{T'}(u) + (n-1).$$

Складывая с $W(T')$, получаем требуемую формулу. □

Пример 2. Рассмотрим дерево T на n вершинах, полученное из звезды S_{n-1} добавлением ещё одной вершины v, соединённой с листом u звезды S_{n-1}. По Теореме 2 имеем

$$W(T) = W(S_{n-1}) + d_{S_{n-1}}(u) + n - 1.$$

Для звезды S_{n-1} известно, что

$$W(S_{n-1}) = (n-2)^2,$$

а для листа u: расстояние до центра равно 1, а до остальных $n-3$ листьев равно 2, поэтому

$$d_{S_{n-1}}(u) = 1 + 2(n-3) = 2n-5.$$

Следовательно,

$$W(T) = (n-2)^2 + (2n-5) + n - 1 = n^2 - n - 2.$$

Так как в доказательстве того, что T является деревом, использовался лишь подсчёт числа пар вершин $\{v,y\}$, очевидно, что Теорему 2 можно обобщить и на связные графы. Поэтому отдельно запишем обобщенную теорему.

Теорема 3. *Пусть G — связный граф, а $v \in V(G)$ — лист (вершина степени 1). Пусть также $(v,u) \in E(G)$ и $G' = G - v$ — подграф графа G, полученный удалением вершины v (и инцидентного ей ребра). Тогда*

$$W(G) = W(G') + d_{G'}(u) + |V(G')|.$$

Теорема 4. *Пусть T — дерево порядка $n \geq 2$, полученное следующим образом: берём вершину v и поддеревья $T_1, \ldots, T_m$, попарно не пересекающиеся по вершинам, и соединяем v с вершиной $u_i \in V(T_i)$ для каждого $i = 1, \ldots, m$. Тогда*

$$W(T) = \sum_{i=1}^{m} \Bigl[W(T_i) + (n - |V(T_i)|)\, d_{T_i}(u_i) - |V(T_i)|^2 \Bigr] + n(n-1). \tag{2}$$

Доказательство. Зафиксируем i и рассмотрим пары вершин (x,y), где $x \in V(T_i)$.

Случай 1: $y \in V(T_i)$. Тогда $d_T(x,y) = d_{T_i}(x,y)$, а сумма по всем таким парам равна $W(T_i)$.

Случай 2: $y \in V(T_j)$, $j \neq i$, или $y = v$. Пусть $y \in V(T_j)$, $j \neq i$ (случай $y = v$ получается аналогично и включится в итоговую формулу). Единственный путь в дереве проходит через вершины u_i, затем v, затем u_j, поэтому

$$d_T(x,y) = d_{T_i}(x,u_i) + d_T(u_i,v) + d_T(v,u_j) + d_{T_j}(u_j,y).$$

Поскольку u_i соединена с v одним ребром, имеем $d_T(u_i,v) = 1$ и $d_T(v,u_j) = 1$, откуда

$$d_T(x,y) = d_{T_i}(x,u_i) + 2 + d_{T_j}(u_j,y).$$

Суммируя по всем $y \notin V(T_i)$, заметим, что для фиксированного $x \in V(T_i)$ величина $d_{T_i}(x, u_i)$ повторяется ровно $n - |V(T_i)|$ раз, поэтому

$$\sum_{y \notin V(T_i)} d_T(x, y) = (n - |V(T_i)|)\, d_{T_i}(x, u_i) + \sum_{y \notin V(T_i)} \big(2 + d_{(\text{в своей компоненте})}\big).$$

После суммирования по всем $x \in V(T_i)$ и приведения подобных членов получаем (в стандартной записи через $d_{T_i}(u_i) = \sum_{x \in V(T_i)} d_{T_i}(x, u_i)$) цепочку преобразований:

$$\begin{aligned}
W(T) &= \sum_{i=1}^{m} \left[W(T_i) + \frac{1}{2} \sum_{x \in V(T_i)} \sum_{\substack{y \in V(T) \\ y \notin V(T_i)}} \big(d_{T_i}(x, u_i) + d_T(u_i, v) \big) \right] \\
&= \sum_{i=1}^{m} \left[W(T_i) + \sum_{x \in V(T_i)} \big(d_{T_i}(x, u_i) + 1 \big)\, (n - |V(T_i)|) \right] \\
&= \sum_{i=1}^{m} \Big[W(T_i) + (n - |V(T_i)|) \big(d_{T_i}(u_i) + |V(T_i)| \big) \Big] \\
&= \sum_{i=1}^{m} \Big[W(T_i) + (n - |V(T_i)|)\, d_{T_i}(u_i) - |V(T_i)|^2 \Big] \;+\; n \sum_{i=1}^{m} |V(T_i)|.
\end{aligned}$$

Так как вершины $T_1, \ldots, T_m$ вместе дают все вершины дерева, кроме v, имеем

$$\sum_{i=1}^{m} |V(T_i)| = n - 1,$$

и потому последний член равен $n(n-1)$. Получаем формулу (2). □

Пример 3. Рассмотрим дерево T на n вершинах, полученное следующим образом: берём путь P_{l+1} и к одному из его концов v присоединяем ещё $n - l - 1$ висячих вершин (то есть добавляем $n-l-1$ листьев, смежных с v). Покажем, как вычислить $W(T)$ с помощью Теоремы 4.

В качестве разделяющей вершины возьмём вершину v. Тогда поддеревья T_i имеют вид:

- $n-l-1$ поддеревьев, состоящих из одной вершины (лист), для каждого из них $|V(T_i)| = 1$, $W(T_i) = 0$, $d_{T_i}(u_i) = 0$;
- одно поддерево, являющееся путём на l вершинах (это P_l), причём u — его конец, смежный с v.

Для одиночной вершины вклад в сумму из (2) равен

$$W(T_i) + (n - |V(T_i)|) d_{T_i}(u_i) - |V(T_i)|^2 = 0 + (n-1) \cdot 0 - 1^2 = -1,$$

поэтому суммарный вклад всех $n - l - 1$ листьев равен $-(n - l - 1)$.

Для пути P_l используем известные формулы:

$$W(P_l) = \binom{l+1}{3}, \qquad d_{P_l}(u) = \sum_{k=0}^{l-1} k = \binom{l}{2}, \qquad |V(P_l)| = l.$$

Тогда вклад этого поддерева в (2) равен

$$W(P_l) + (n-l)\, d_{P_l}(u) - l^2 = \binom{l+1}{3} + (n-l)\binom{l}{2} - l^2.$$

С учётом последнего слагаемого $n(n-1)$ из (2) получаем

$$\begin{aligned} W(T) &= \left(\binom{l+1}{3} + (n-l)\binom{l}{2} - l^2\right) \; - \; (n-l-1) \; + \; n(n-1) \\ &= \binom{l+1}{3} + (n-l)\binom{l}{2} - (n-l-1) - l^2 + n(n-1). \end{aligned}$$

Упрощая выражение, получаем компактную форму:

$$W(T) = \binom{l}{3} + (n-l-1)\binom{l}{2} + (n-1)^2.$$

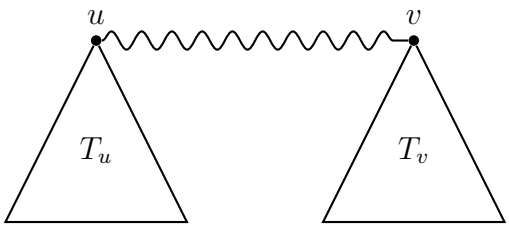


Рис. 3: Деревья T_u и T_v, соединённые путём из k новых вершин.

4.3 Итог

В этой главе мы показали основные прямые и итерационные методы вычисления индекса Винера в деревьях. С помошью итерационных методов вычислять индекс Винера в растущих деревьях можно с помощью нехитрых формул, что минимизирует процесс вычислений и делает их максимально быстрыми.

5 Теоретические основы итерационного вычисления индекса Винера на растущем графе

Была поставлена задача реализовать конкретную процедуру по вычислению индексов Винера для растущих графов эффективнее, чем в готовом пакете networkX. В пакете существует только одна функция вычисления индекса Винера, которая реализована для статичного графа. Поэтому вычисления с помощью готовой функции будут крайне не эффективными. Следовательно логично реализовать ее самим.
Для начала надо понять как это можно сделать для случая `leaf` и `edge`.

5.1 Добавление `leaf`

Рассмотрим операцию роста `leaf`: к графу G добавляется новая вершина $x \notin V$ и ровно одно ребро (x, p), где $p \in V$ — уже существующая вершина. Обозначим новый граф через

$$G' = G + \{x\} + \{(x, p)\}.$$

Добавление листа не изменяет расстояния между старыми вершинами, так как возникнуть новых более коротких маршрутов не может. Следовательно, изменения индекса Винера возникают только за счёт новых пар (x, v), $v \in V$:

$$W(G') = W(G) + \sum_{v \in V} d_{G'}(x, v). \tag{3}$$

Так как граф невзвешенный, величина $\sum_{v \in V} d_{G'}(x, v)$ вычисляется одним обходом в ширину (BFS) из источника x за время $O(|V| + |E|)$ на текущем графе.

5.2 Добавление `edge`

Рассмотрим операцию роста `edge`: добавляется ребро между двумя уже существующими вершинами $a, b \in V$:

$$G' = G + \{(a, b)\}.$$

Добавление ребра может сократить кратчайшие пути для большого числа пар вершин. Поэтому точное итерационное обновление $W(G)$ при вставках рёбер неочевидно; поэтому потребуется точный пересчёт $W(G')$.

5.3 Общая схема модулей

Реализация разделена на два файла:

- `wiener.py` — класс `WienerGrowingUnweighted`, вычисление индекса Винера при росте графа;
- `main.py` — загрузка датасета, построение графа, рандомизированное выполнение операций роста, измерение времени и сохранение графика.

5.4 Программная реализация на python

Реализуем Класс и соотвествующие ему функции.

```
class WienerGrowingUnweighted:


        def __init__(self) -> None:
                self.G = nx.Graph()
                self.W = 0.0

        def add_initial_node(self, u: Any) -> None:
                self.G.add_node(u)
                self.W = 0.0

        def add_leaf(self, u: Any, attach_to: Any) -> float:
                self.G.add_node(u)
                self.G.add_edge(u, attach_to)
                dist = nx.single_source_shortest_path_length(self.G, u)
                self.W += sum(dist.values())
                return self.W

        def add_edge(self, a: Any, b: Any) -> float:
                if a == b:
                        return self.W

                self.G.add_edge(a, b)
                self.W = float(nx.wiener_index(self.G))
                return self.W
```

5.5 Устройство кода генерации растущего графа

Далее часть `main.py`, отвечающая за генерацию *растущего графа*.

На вход подаётся файл со списком рёбер (формат *edgelist*). Функция загрузки читает файл в байтовом режиме (без декодирования в UTF-8), формируя неориентированный граф G (объект `networkx.Graph`).

Если исходный граф несвязен, выбирается крупнейшая связная компонента H:

$$H \;=\; \mathrm{LargestConnectedComponent}(G).$$

Дальнейший рост происходит так, чтобы текущий граф на каждом шаге оставался связным.

Функция `bfs_tree_edges_sorted(H, root)` строит BFS-остов из вершины *root*. Результатом является список рёбер BFS-дерева в порядке открытия:

$$T = \big((p_1, c_1), (p_2, c_2), \ldots, (p_{n-1}, c_{n-1})\big),$$

где каждое ребро (p_i, c_i) означает: вершина c_i впервые обнаружена из p_i.

На основе T строится отображение

$$\mathrm{children\_by\_parent}[p] = \{\, c : \; (p, c) \in T \,\},$$

которое в коде реализовано как словарь `parent` $\rightarrow$ `deque(children)`. Эта структура позволяет добавлять новые вершины только тогда, когда их родитель уже присутствует в текущем графе.

5.5.1 Дополнительные рёбра (не входящие в BFS-остов)

Все рёбра исходного графа H, не входящие в BFS-остов, образуют множество *дополнительных рёбер*:

$$E_{\text{extra}} \;=\; E(H) \setminus E(T).$$

В коде это формируется в виде списка `extra_edges` (с нормализацией порядка концов ребра и сортировкой), чтобы вход бы определен.

5.5.2 Почему все выполняется корректно

Для корректного роста вводится множество активных вершин A — вершин, которые уже добавлены в текущий граф роста. В начале:

$$A = \{root\}.$$

Операция `leaf`(c, p) выполнима только если $p \in A$. Операция `edge`(u, v) выполнима только если $u \in A$ и $v \in A$.

Чтобы эффективно находить `edge`-операции, которые становятся выполнимыми после добавления новой вершины, строится индекс инцидентности:

$$\text{incident}[x] = \{\, i \mid \text{extra\_edges}[i] \text{ инцидентно вершине } x \,\}.$$

После добавления вершины x активируются все рёбра (x, y), для которых $y \in A$. В коде это множество хранится как `eligible_edges` (индексы в `extra_edges`).

5.5.3 Рандомизированное чередование `leaf`/`edge`

Функция `make_randomized_ops(...)` строит последовательность шагов роста (до `max_ops` шагов), случайно чередуя `leaf` и `edge` при соблюдении всех условий.

Если доступны оба типа операций, то `edge` выбирается с вероятностью p_{edge}, иначе выбирается `leaf`. Если доступен только один тип, выбирается он.

Роль seed. Для воспроизводимости используется генератор `random.Random(seed)`. При фиксированном `seed` последовательность случайных выборов повторяется, то есть траектория роста (порядок добавления вершин и рёбер) будет одинаковой при каждом запуске.

5.5.4 Итог

Итоговая логика генерации роста:

1. загрузить граф G и выделить крупнейшую компоненту H;
2. выбрать корень $root$;
3. построить BFS-остов T и структуру `children_by_parent`;
4. построить список дополнительных рёбер `extra_edges`;
5. начиная с $A = \{root\}$, итеративно выполнять шаги:
   - `leaf`: добавить новую вершину из `children_by_parent` и обновить A;

- после добавления вершины активировать допустимые дополнительные рёбра в `eligible_edges`;
- `edge`: выбрать одно допустимое ребро из `eligible_edges` и добавить его;

пока не выполнено `max_ops` шагов или пока операции не исчерпаны.

Такая конструкция обеспечивает, что на каждом шаге строится корректный промежуточный граф (все рёбра добавляются только между уже существующими вершинами), при этом порядок роста может быть случайно перемешан, но остаётся воспроизводимым при фиксированном seed.

6 Резульаты

Написанная мой программа была протестированна на трех датасетах (), взятых из Stanford Large Network Dataset Collection(SNAP). Результат оказался предсказуемым, времени потребовалась почти в 2 раза меньше, чем у стандартной функции networkX. Даже чисто интуитивно можно заметить, что ветвь синяя линия напоминает ветвь параболы (функция $f(x) = x^2$), а оранжевая $f(x) = \frac{x^2}{2}$

Графики

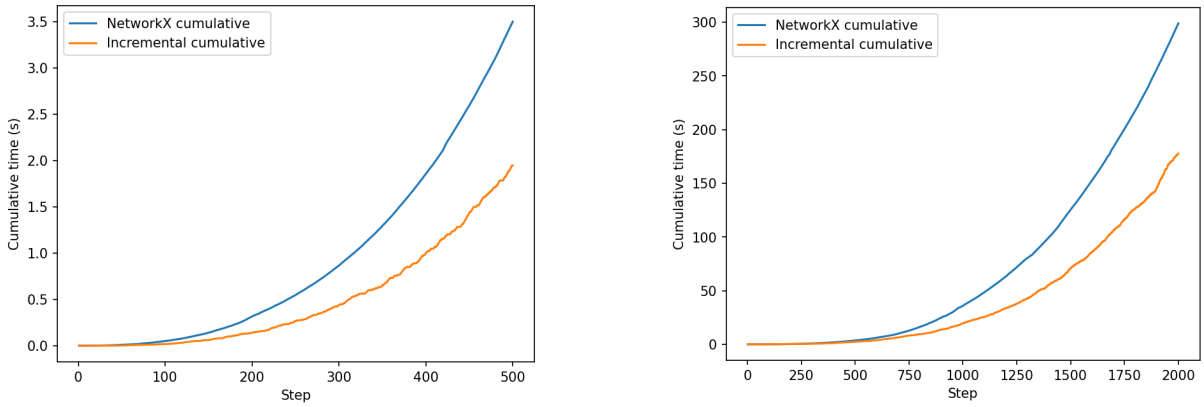


Рис. 4: Сеть сотрудничества Arxiv в области физики высоких энергий

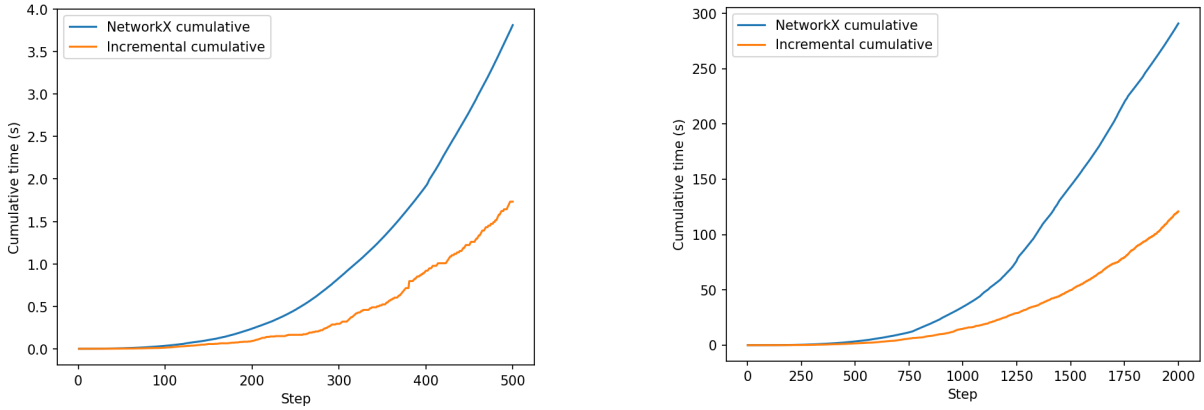


Рис. 5: Сеть электронной переписки компании Enron

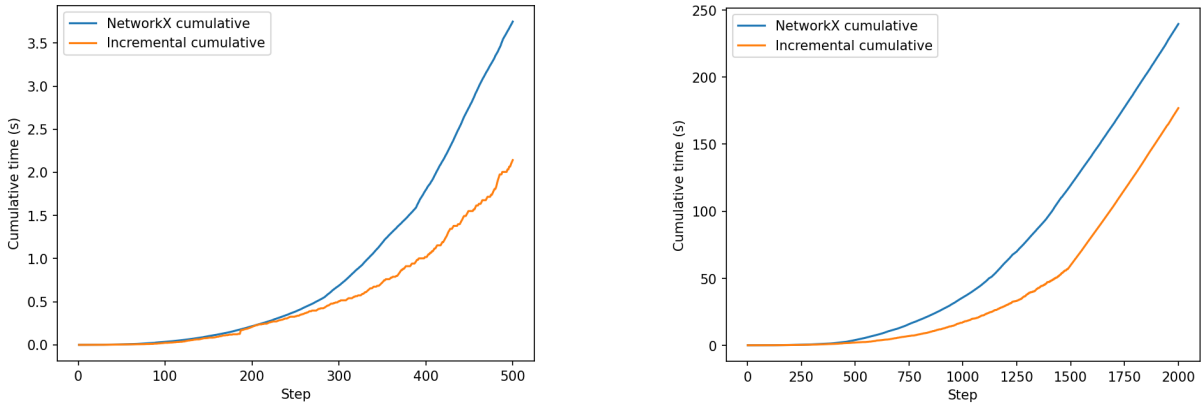


Рис. 6: Социальные круги из Facebook (анонимизированные)

7 Вывод

В данной работе я рассмотрел различные способы вычисления индекса Винера для растущего графа. Были предложены аналитические решения некоторых случаев роста графа. Было проанализирован частный случай дерева и усовершенствована функция вычисления индекса Винера для растущих графов из библиотеки networkX и предоставлены графики скорости вычислений.

8 Дальнейшие направления для развития

Индекс Винера применяется во многих сферах, поэтому было бы логично для будующих исследований испкать новые способы эффективного вычисления, а так же расширять и обобщать рекуррентные формулы на более широкий класс графов, не ограничиваясь деревьями.
Так же интерес могут представлять исследование погрешностей при приближённом обновлении индекса Винера — оценка, насколько точно можно обновлять $W(G)$ на больших графах, жертвуя частью точности ради выигрыша в скорости.
Наконец, полученные алгоритмы и подходы можно применять к реальным задачам: от анализа сетей различной природы (социальных, транспортных, коммуникационных) до исследования молекулярных структур.

9 Список используемой литературы

[1] A. Altassan и H. Ahmad. “Computation of Distance-Based Polynomials and Topological Indices for Generalized Friendship Graphs (G (n, k)): A Python-Based Approach”. В: *Malaysian Journal of Mathematical Sciences* 19.4 (2025), с. 1453—1470. DOI: 10.47836/mjms.19.4.15.

[2] Bojan Mohar и Tomaž Pisanski. “How to compute the Wiener index of a graph”. В: *Journal of Mathematical Chemistry* 2 (1988), с. 267—277. DOI: 10.1007/BF01167206.

[3] *Индекс Винера*. Википедия. URL: https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_%D0%92%D0%B8%D0%BD%D0%B5%D1%80%D0%B0.

[4] Aric A. Hagberg, Daniel A. Schult и Pieter J. Swart. “Exploring network structure, dynamics, and function using NetworkX”. В: *Proceedings of the 7th Python in Science Conference (SciPy 2008)*. Под ред. Gaël Varoquaux, Travis Vaught и Jarrod Millman. Pasadena, CA, USA, авг. 2008, с. 11—15.

[5] *NetworkX: Software for Complex Networks*. Официальная документация. URL: `https://networkx.org/en/`.