Tablice sufiksów i drzewa sufiksów

dr inż. Marcin Ciura
mgc@agh.edu.pl

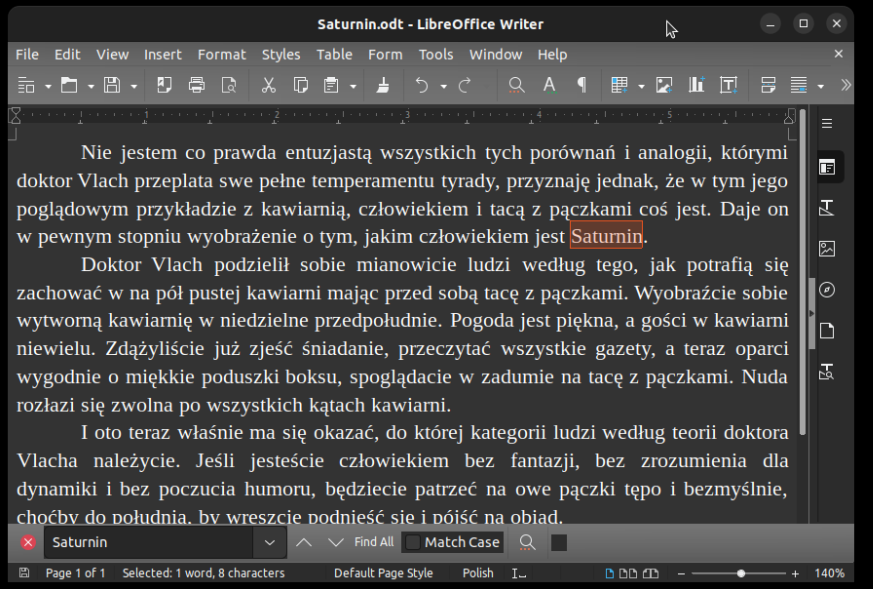
Wydział Informatyki, Akademia Górniczo-Hutnicza

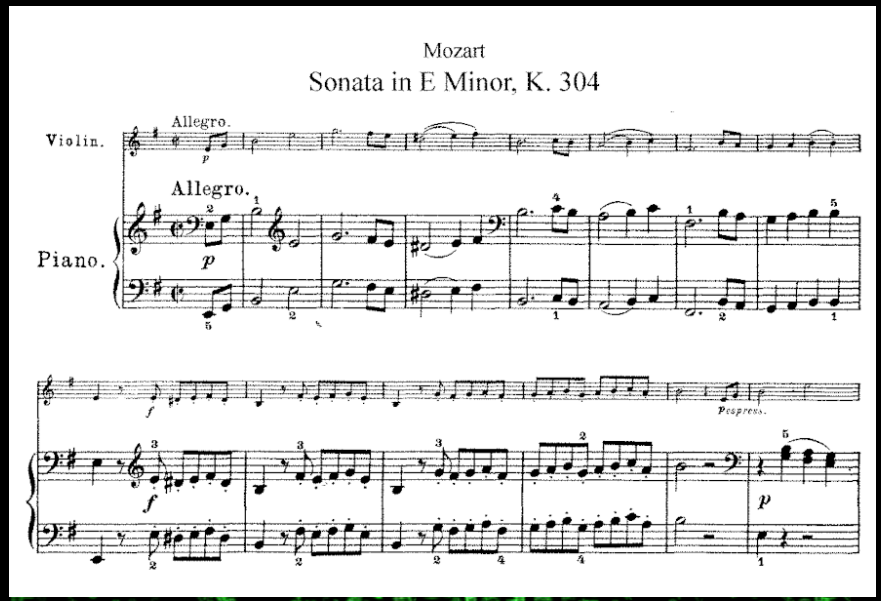
Plan wykładu

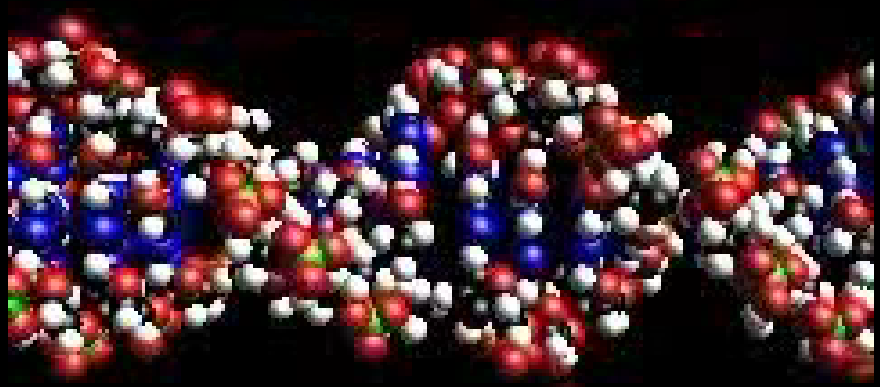
- Tablice sufiksów
- Zastosowania tablic sufiksów
- Drzewa sufiksów
- Zastosowania drzew sufiksów
- Algorytm Ukkonena budowania drzew sufiksów

Po co nam tablice sufiksów i drzewa sufiksów?

Żeby wyszukiwać wzorce w tekstach, i nie tylko po to :-)







Prehistoria

Konkordancje

Aaron with sons
Ex. 40:31. M. and A. & s. washed thereat
Le. 1: 7. s. of A. shall put fire upon altar
3:13. s. of A. sh. sprinkle blood on alt.
6: 9. Command A. and s. law of burnt-off.
14. law of meal-off. s. of A. sh. offer
16. wh. is left shall A. and s. eat
20. This is oblation of A. and s.
25. Speak unto A. and s. (17:2; 22:2; 22:18; Nu. 6:23)
7:10. Every meal-off. sh. s. of A. have
33. s. of A. have right thigh (34)
35. the anoint-portion of A. and s.
8: 2. Take A. and s. and the garments
6. M. bro't A. and s., & washed them
14. A. and s. laid hands on bullock
22. A. and s. laid hands upon ram
30. M. sprinkled (blood) up. A. and s. sanctified A., garments and s.
31. as I com'ded, A. and s. sh. eat it
36. A. and s. did all JH. commanded
9: 1. Moses called A. and his s., and
9. s. of A. presents the blood
10: 1. s. of A., took each his censer
6. M. said unto A., his s., Let not
12. M. spake unto A., his s., Take
16. with Eleazar & Ithamar, s. of A.
13: 2. leprosy, brought unto A. or s.
16: 1. after death of two s. of A.
21: 1. Speak unto s. of A. none defile
24. So M. spake unto A. and s.
24: 9. (bread) shall be for A. and s.
Nu. 3: 2. the names of the s. of A. (4) [19]
9. give Levites unto A. and his s.
10. thou shalt appoint A. and s.
32. Eleazar, s. of A. (Jos. 24:33; Jo. 20:28; Ezr. 7:5)
38. encamp eastward sh. be A. and s.
48. give money unto A. and s.
51. M. gave money unto A. and s.
4: 5. A. and s. shall take down veil
15. A. and s. made end of covering
19. sh'gs of Eleaz., s. of A. sh. be ol
28. A. and s. shall appoint service
27. At com'dment of A. & s. be service
28. charge under Ithamar s. of A. (33)
6:23. Speak unto A. & s., ye sh. bless
7: 8. service under Ithamar, s. of A.
8:22. went Lev. to service bef. A. & s.
10: 8. s. of A. shall blow trumpets
16:37. Eleazar s. of A., take censer
40. J.H. said un. A. Thou & thy s. (5)
25: 7. Phinehas, s. of El. s. of A.
11. s. of A. hath turned my wrath
1 Ch. 6: 3. s. of A., Nadab, Abihu, etc. (50:1)
49. A. and s. offered upon altar
50. these are the s. of A.
54. dwelling-places of s. of A.
57. s. of A. gave cities of refuge
23: 4. David gathered s. of A.
13. A. was separated, & s. for ever
28. their office to wait on s. of A.
32. should keep charge of s. of A.
24: 1. courses of s. of A. were these
31. cast lots even as s. of A.
2 Ch. 13: 9. have ye not driven out s. of A.
10. we have priests, s. of A.
26:18. s. of A. that are consecrated
29:21. commanded s. of A. to offer
31:19. s. of A. that were in the fields
35:14. s. of A. were busied in offering
Levites prepared for s. of A.
Ne. 12:47. yet apart wh. was for s. of A.

AARON'S (Aar'on's, or Aa'ron's)
Ex. 6:25. Eleazar, A' son took wife
7:12. A' rod swallowed up their rods
28:30. they shall be upon A' heart
38. it shall be upon A' forehead
40. for A' sons thou shalt make coats
29:26. breast of A' ram of consecra.
Le. 1: 5. A' sons sh. sprinkle blood (11; 3:2)
8. A' sons sh. lay pieces in order
2: 2. sh. bring (meal-off.) to A' sons
3. left of the meal-off. be A' sons
8: 5. A' sons sh. burn it on altar
8:12. he poured oil upon A' head
13. Mos. bro't A' sons & clothed them
9:21. A' sons offer'd un. him blood
10:16. Moses was angry with A' sons
Nu. 17: 3. write A' name upon rod of Levi
Ps. 133: 2. like precious oil upon A' beard
Ho. 9: 4. pot holding A' rod that budded

ABADDON (A-bād'don)—*destruction, ruin.*
A town synagogues with death and Sheol.
Jb. 26: 6. And A. hath no covering
Pr. 15:11. Sheol and A. are before JH.
27:20. Sheol and A. are nev. satisfied
Re. 9:11. angel of the abyss is A.

ABAGTHA (A-bag'tha)—A Persian name signifying *happy, prosperous.*
Es. 1:10. A., etc. seven chamberlains

ABANAH (Ab'a-nah, or A-ba'nah)—*probably, stony.*
Presumably the modern Barada, the more important of the two rivers of Damascus. It rises in Anti-Lebanon, runs along the north wall of the city, and loses itself in the swampy Meadow Lakes.
2 K. 5:12. A. and Pharpar, rivers of D.

ABARIM (Ab'a-rim, or A-ba'rim)—*those beyond, i.e. beyond the Jordan.*
The mountain range east of the Jordan, the highest peak of which is the Nebo, from which Moses viewed the promised land (De. 34:1-4).
Nu. 27:12. Get thee in. mt. of A. (De. 32:49)
33:47. encampment in mountains of A.
48. Journeyed fr. mountains of A.
Je. 22:20. *cry* from A. for all thy lovers

ABASE
Jb. 40:11. Look upon proud, and a.
Is. 2:11. A. of lofty eyes of man shall be a.
Eze. 21:26. a. that which is high
Da. 4:37. walk in pride able to a.

ABASED
Ph. 4:12. I know how to be a.

ABASING
1 Co. 11: 7. did I sin in a. myself?

ABATED
Ge. 8: 8. dove to see if waters a.
11. Noah knew waters were a.
Nu. 11: 2. prayed unto JH., & the fire a.
De. 34: 7. nor his natural force a.
Jo. 8: 4. Then their anger was a.

ABATEMENT
Le. 25:16. an a. shall be made

ABBA (Ab'ba)—Aramaic word for *father.*
Mk. 14:36. A, Father, all things possible
Ro. 8:15. whereby we cry A, Father.
Ga. 4: 6. Spirit, crying A, Father

ABDA (Ab'da)—Aramaic for *servant.*
1. **Father of Adoniram, Solomon's tribute-officer.** [task-work.
1 K. 4: 6. son of A. was over men subject to
2. **The son of Shammua, called Obadiah in 1 Ch. 9:16.**
Ne. 11:17. A. son of Shammua, son of Galal

ABDEEL (Ab'de-el)—*servant of God.*
The father of Shelemiah, one of the men ordered by King Jehoiakim to arrest Jeremiah.
Je. 36:26. king commanded Shelemiah son of A.

ABDI (Ab'di)—*servant of* (God).
1. **A Levite of the family of Merari.**
1 Ch. 6:44. Ethan son of Kishi, son of A.
2 Ch. 29:12. sons of Merari, Kish son of A.
2. **One of the family of Elam who divorced his foreign wife.**
Ezr. 10:26. and A., and Jeremoth, and Elijah.

ABDIEL (Ab'di-el)—*servant of God.*
A Gadite, head of a father's house.
1 Ch. 5:15. Ahi the son of A., son of Guni

ABDON (Ab'don)—*servile.*
1. **One of the judges of Israel.**
Ju. 12:13. A. son of Hillel judged Is. (15)
2. **A Benjamite, head of father's house, family of Elnashah.**
1 Ch. 8:23. and A. and Zichri, and Hanan
3. **A Benhamite, first-born of Jeiel of Gibeon.**
1 Ch. 8:30. his first-born son A., & Zur (9:36)
4. **One sent by king Josiah to inquire of Jehovah, called Achbor son of Micaiah in 2 K. 22:12.**
2 Ch. 34:20. king commanded A. son of Micah

ABDON (Ab'don)
A town in the territory of Asher.
Jos. 21:30. A. with its suburbs (1 Ch. 6:74)

ABEDNEGO (A-bēd'ne-gō)—*servant of Nego.*
The name given by the prince of the eunuchs to Azariah, one of the four young princes of Judah, deported to Babylon by Nebuchadnezzar.
Da. 1: 7. and to Azariah of A. [and A.
7. he appointed Shadrach, Meshach,
3:12. had appointed Shad., Mesh. & A.
13. commanded to bring S., M. and A.
14. Is it of purpose, O S., M. & A.?
16. Shadrach, Meshach & A. answered
19. visage was changed ag'n S., M. & A.
20. to bind Shadrach, Meshach & A.
22. fire slew men took up S., M. & A.
Da. 3:23. Shadrach, M. & A. fell down bound
26. and said, S., M. & A., ye servants
Then S., M. & A. came forth out of
28. Blessed be the God of S., M. & A.
30. king promoted S., M. & A. in the

ABEL (A'bel)—*formerly interpreted breath, but now thought to be ox.*
Second son of Adam.
Ge. 4: 2. again she bare A.
A. was a keeper of sheep
4. A. brought firstlings of flock
Jehovah had respect unto A.
8. Cain told A. his brother
Cain rose up ag. A. & slew him
9. Jehovah said, Where is A.?
25. another seed instead of A.
Mt. 23:35. from the blood of A. (Lu. 11:51)
He. 11: 4. By faith A. off'd more ex. sac.
12:24. blood speaketh better than A.

ABEL (A'bel)—*a grassy place, meadow.*
A fortress in the most northerly part of Naphtali, identified with the modern village of Abel, a few miles above Lake Huleh.
2 S. 20:18. went unto A.
15. besieged him in A.
18. shall ask counsel at A.

ABEL-BETH-MAACAH (A'bel-bēth-mā'a-cah)—*meadow near Beth-maacah.*
A fortified town in Naphtali, captured by Benhadad at the instance of Asa, along with Ijon and Dan.
1 K. 15:20. Ben sente Ijon, etc. and A.
2 K. 15:29. Tiglath-pil. took Ijon and A.

ABELCHERAMIM (A'bel-chēr'a-mīm)—*meadow of vineyards.*
A locality east of the Jordan, a few miles from Rabbah.
Ju. 11:33. even twenty cities, and un. A.

ABEL-MAIM (A'bel-mā'im)—*meadow of the waters.*
Apparently another name for Abel-beth-maacah (1 K. 15:20).
2 Ch. 16: 4. Ben-hadad sent against A.

ABEL-MEHOLAH (A'bel-me-hō'lah)—*meadow of dancing.*
A town in Issachar, in which the prophet Elisha was born, ten or twelve miles south of Beth-shean.
Ju. 7:22. fled as far as A.
1 K. 4:12. from Beth-shean to A.
19:16. Elisha son of Shaphat of A.

ABEL-MIZRAIM (A'bel-miz'ra-im, or A'bel-mīz'ra-ĭm)—*meadow of Egypt.*
Ge. 50:11. wherefore it was called A.

ABEL-SHITTIM (A'bel-shit'tim)—*meadow of acacias.*
A locality in the lowlands of Moab where the Israelites encamped.
Nu. 33:49. encamped even unto A.

ABHOR
Le. 26:11. my soul shall not a. you
15. if your soul a. mine ord'ns.
30. my soul shall a. you
44. neither will I a. them
De. 7:26. an abomination shall a.
23: 7. not a. an Edomite: an Egyptian
1 S. 27:12. made Israel a. him
Jb. 9:31. mine own clothes shall a. me
19:19. my familiar friends a. me
30:10. a. to spit before my face
42: 6. Wherefore I a. myself
Ps. 119:163. I hate and a. falsehood
Pr. 24:24. nations shall a. him
Je. 14:21. Do not a. us for thy name's
Am. 5:10. they a. him speaketh uprightly
6: 8. I a. the excellency of Jacob
Mi. 3: 9. Hear this, that a. justice
Ro. 12: 9. A. that which is evil

ABHORRED
Ex. 5:21. ye have made our savor a.
Le. 26:43. therefore have I a. them
26:43. their soul a. my statutes
Dt. 32:19. JH. saw it, and a. them
1 S. 16:21. thou art a. of thy father
1 K. 11:25. (Rezon) a. Israel
Ps. 78:59. nor a. affection of afflicted
89:38. When God heard a. Israel
89:40. hast a. covenant of thy serv't
106:40. JH. a. his inheritance
Pr. 22:14. He that is a. of JH.
Is. 22: 7. JH. built a. his sanctuary
Eze. 16: 5. for thy person was a., in day

ABHORRENCE
Da. 12:2(m). to shame and *everlasting* a.
Ro. 2:22. thou that a. idols, dost thou

ABHORREST
Is. 7:16. land whose two kings thou a.

ABHORRETH
Jb. 33:20. so that his life a. bread
Ps. 5: 6. JH. a. the blood-thirsty
36: 4. he a. not evil
107:18. their soul a. all food
Is. 49: 7. to him whom the nation a.

ABHORRING
Is. 66:24. they shall be a. unto all

ABI (A'bi)—*Jehovah is father.*
Daughter of Zechariah, and wife of Ahaz.
2 K. 18: 2. (Hezekiah's) mother's name was A.

ABI-ALBON (A'bi-al'bon)—*father of strength.*
Called Abiel in 1 Ch. 11:32.
2 S. 23:31. one of thirty, A. etc.

ABIASAPH (A-bi'a-săph)—*the father gathers.*
A descendant of Levi, *family of Kohath.* [A.
Ex. 6:24. sons of Korah: Assir, & Elkanah, &
ABIATHAR (A-bi'a-thăr)—*father of abundance.*
Eleventh high priest in succession from Aaron.
1 S. 22:20. A. fled to Dav. David (23:6)
21. A. told Dav. Saul slain priest
22. David said unto A., I knew
23: 6. when A. fled to David
9. David said to A., bring ephod
30: 7. A. brought hither the ephod
2 S. 8:17. Ahimelech son of A. (1 Ch. 24:6)
15:24. A. went up, until all people
27. Jonathan son of A. (36; 1 K. 1:42)
29. Zadok & A. carried ark of God
35. shalt tell it to Zadok & A.
17:15. Hushai said unto Zadok and A.
19:11. David sent to Zadok and A.
20:25. Zadok and A. were priests (1 K. 4:4; 1 Ch. 18:16)
1 K. 1: 7. (Adonijah) conferred with A.
19. (Adonijah) hath called A. (25)
42. Jonathan the son of A.
2:22. ask for him, for A.
26. unto A. Anat, worthy of death
27. Solomon thrust A. from priest
35. Zadok did king put in room of A.
1 Ch. 15:11. David called for Zadok & A.
27:34. after Ahithophel was A.
Mk. 2:26. entered house of God when A.

ABIATHAR'S (A-bi'a-thar'a)
2 S. 15:35. Zadok's son, & Jonathan A' son

ABIB (A'bib)
Ex. 13: 4. ye go forth in month A.
23:15. feast of unleavened bread in month
34:18. (Ex. 13:4; De. 16:1)

ABIDA (A-bi'da)—*father of knowledge.*
A descendant of Abraham, and ancestral head of a clan of Midian.
Ge. 25: 4. sons of Midian, Ephah, A., etc.
1 Ch. 1:33. sons of Midian, Ephah, A., etc.

ABIDAN (Ab'i-dan, or A-bi'dan)—*father is judge.*
A Benjamite, son of Gideoni, appointed along with others out of every tribe to number Israel.
Nu. 1:11. tribe of Benjamin) A.
2:22. prince of Benjamin sh. be A.
7:60. A. his oblation was (65)
10:24. over host of Benjamin was A.

ABIDE
(Shall abide, To abide)
Ge. 19: 2. we will a. in the street
22: 5. a. here with the ass
24:55. Let the damsel a. with us
29:19. I give her to thee; a. with me.
24:33. let serv't a. instead of lad
Ex. 10: 9. ye every man in his place
Nu. 22: 5. people fr. Egypt a. over az. me
32:23. every thing that may a. fire
Ru. 1: 8. but a. here fast by maidens.
1 S. 1:22. before JH. & there a. for ever.
19: 2. a. in a secret place, and hide
22: 5. unto David, A. not in stronghold
23. A. thou with me, fear not: for
2 S. 11:11. ark, and Israel a. in booths
15:20. thereto have I a. them
16:18. his will I be, & with him a.
2 K. 14:10. glory thereof, & a. at home
24:19. a. now at home; why shd'st?
2 Ch. 32:10. Whereon trust, that ye a. siege?
Jb. 24:13. the light. Nor a. in paths th'of
38:40. (lions) a. in the covert
39: 9. Will wild-ox . . . a. by thy crib?
Ps. 102:12. But thou, O JH., wilt a. for ev.
119:152. Thou a. this day ac. to thine
(91). *As for thine ord'ns* they a.
Pr. 7:11. her feet a. not in her house
Je. 42:10. If ye will still a. in this land
Ho. 3: 3. Thou shalt a. for me many days
Joe. 2:11. day of JH. . . . who can a. it?
Na. 1: 6. can a. fierceness of his anger?
Mal. 3: 2. Who can a. day of his coming?
Mt. 10:11. there a. till ye go forth (Mk. 6:10; Lu. 9:4)
Mk. 14:34. a. ye here, and watch
Lu. 19: 5. to-day I must a. at thy house.
24:29. A. with us; for it toward evening
Jn. 8:35. If ye a. in my word, then are ye
12:46. believeth, may not a. in darkness
15: 4. A. in me, and I in you. As branch except it a. in the vine; so
not. can ye, except ye a. in me.
6. man a. not in me, he is cast forth
7. ye a. in me, & my words a. in you
9. have loved you: a. ye in my love.
10. my Fa's com's, & a. in his love.
16. and that your fruit should a.
1 Jn. 2:6. come into my house, and a.
20:23. that bonds & afflictions a. me
27:31. Except these a. in the ship, ye
1 Co. 7: 8. good . . . if they a. even as I
20. Let each man a. in that calling
24. wherein he was called, th'in a.
40. But she is happier if she a.
Ph. 1:25. I shall a., yea, & a. with you all
2 Ti. 3:14. a. thou in things hast learned
1 Jn. 2:24. let that a. in you, ye heard that which ye heard . . . a. in you
27. as it taught you, ye a. in him
28. now, little children, a. in him
3:17. how dwells love of God a. in him?
4:13. Hereby we know that we a. in him
Shall abide
Le. 8:35. at door of tent of meeting a. ye a.
25:15. wages of servant a. not a.
De. 3:19. your wives a. a. in your cities
Jos. 2:14. a. a. in land which Moses gave
18: 5. Judah a. a. . . . on the south
Joseph a. a. . . . on the north.
1 S. 2: 7. ark of God a. not a. with us
Ps. 61: 7. His a. is, before God for ever
63:35. they a. a. there, and have it
91: 1. S. a. under shadow of Almighty
Pr. 15:31. heareneth to reproof s. a.
19:23. *that hath it* a. a. satisfied
Ec. 8:15. for (mirth) a. a. with him in
Is. 32:16. righteousness a. a. in the
18. my people a. a. in a peaceable
Ho. 3: 4. Israel a. a. many days w'out king
Joe. 3:20. But Judah a. a. for ever, and
Mi. 5: 4. a. a.; for now he shall be great
Hab. 2:5(m). hereby man, he a. not a. as
Zec. 5: 4. it a. a. in midst of his house
Jn. 15:10. keep my commandments, ye a. a. in my love
1 Co. 7:24. If any man's work a. a. which
26: 6. be that I a. a., or even winter
Ph. 1:25. that I a. a., and a. with you
To abide
1 S. 30:21. had made t. a. at brook Besor
Je. 10:10. not abide t. a. his indignation
Jn. 1:39. they beheld him t. a. with them
Ac. 15:34(m). *seemed good unto Silas t. a.*
18:20. asked him t. a. a longer time
Ph. 1:24. t. a. in the flesh is more needful

ABIDETH
Le. 5:19. Thou, O Jehovah, a. for ever
Je. 10:17. O thou that a. in the siege
Jn. 1:32. said, Rabbi, where a. thou?

ABIDETH
Nu. 31:23. all that a. not the fire
2 S. 16: 3. Behold, he a. at Jerusalem
Jb. 41:22. In his neck a. strength
Ps. 49:12. man being in honor a. not
55:19. God will hear, even he that a.
91:1(m). *That a. in the secret place*
104:5. established earth, and it a.
125: 1. They that trust in Jehovah a.
Ec. 1: 4. but the earth a. for ever
Je. 22: 9. that a. in city shall die (38:2)
Hag. 2:5(m). And my Spirit a. amo. you
Jn. 3:36. the wrath of God a. on him
6:27. food wh. a. unto eternal life
56. drinketh my blood a. in me
8:35. bondservant a. not, the son a.
12:24. Except grain die, it a. alone
34. We heard, Christ a. for ever
14:17. for he a. with you, & shall be
15: 5. a. in me, beareth much fruit
1 Co. 13:13. now a. faith, hope, love
2 Co. 9: 9. His righteousness a. for ever
2 Ti. 2:13. if faithless, he a. faithful
He. 7: 3. (Melchizedek) a. a priest contin'ly
24. but he, bec. he a. for ever, hath

1 Pe. 1:23. Word of God wh. liveth & a.
25. But word of the Lord a. for ev.
1 Jn. 2: 6. saith he a. in him ought to
10. loveth his bro. a. in light
14. young men, word of God a. in you
17. that doeth the will of God a.
27. anointing of him a. in you
3: 6. Whoso. a. in him sinneth not
9. no sin, bec. his seed a. in him
14. He that loveth not a. in death
24. keepeth his com's a. in him
we know that he a. in us by
4:12. love one ano., God a. in us
15. God a. in him, and he in God
16. he that a. in love a. in God
2 Jn. 2. for truth's sake wh. a. in us
9. a. not in the teaching of Christ
a. in teaching, hath Fa. and Son

ABIDING
Nu. 9:22. tarried upon tabernacle, a. th'on
Ju. 16: 9. lors-in-wait a. in chamber (12)
2 S. 24: 3. his men were a. in innermost
1 Ch. 29:15. our days a shadow, is no a.
Lu. 2: 8. shepherds a. in the field
Jn. 1:33. Spirit descending, & a. up. him
5:38. have not his word a. in you
14:10. Father a. in me doeth works
25. spoken, while yet a. with you.
Ac. 12:23. chamber, where they were a.
He. 10:34. a better possession, & an a.
13:14. we have not here an a. city
1 Jn. 3:15. no murderer hath eter. life a.

ABIDING-PLACES
Jn. 14:2(m). In my Fa's house are many a-

ABIEL (A'bi-el, or A-bi'el)—*father is God.*
1. **Grandfather of Saul.**
1 S. 9: 1. man of Benj. Kish, son of A.
14:51. Ner, fa. of Abner, son of A.
2. **One of David's heroes, evidently the same as Abi-albon (2 S. 23:31).**
1 Ch. 11:32. A. the Arbathite

ABIEZER (A'bi-ē'zer)—*father is help.*
1. **Founder of a family to which Gideon belonged.**
Jos. 17: 2. the lot for the children of A.
1 Ch. 7:18. Hammoleketh bare A.
2. **The clan of that name.**
Ju. 6:34. A. was gathered after (Gideon)
8: 2. gleaning, better than vintage of A.
3. **One of David's thirty valiant men.**
2 S. 23:27. (one of 30) A. the Anathothite
1 Ch. 11:28. (mighty men) A. the Anathothite
27:12. captain for month nin. was A.

ABIEZRITE (A'bi-ēz'rite)
Ju. 6:11. oak pertained unto Joash, the A.

ABIEZRITES (A'bi-ēz'rites)
Ju. 6:24. built altar in Ophrah of the A.
8:32. buried in Ophrah of the A.

ABIGAIL (Ab'i-gāil, or gil)—*father of might.*
1. **Wife of Nabal the Carmelite, who after his death became a wife of David.**
1 S. 25: 3. name of (Nabal's) wife A.
14. one of the young men took A.
18. Then A. made haste
23. And when A. saw David
32. And David said to A. Blessed
36. And A. came to Nabal
39. David sent concerning A.
42. when serv'ts of Dav. came to A.
42. A. went after men pos of Dav.
27: 3. David dwelt at Gath with A.
30: 5. two wives taken captive, A.
2 S. 2: 2. Dav's two wives, Ahinoam & A.
1 Ch. 3: 1. second (born) Daniel of A.
2. **Wife of Jether and mother of Amasa whom David made captain in place of Joab.**
2 S. 17:25. went in to A. dau. of Nahash
1 Ch. 2:16. their sisters were Zeru. & A.
17. And A. bare Amasa

ABIHAIL (Ab'i-hā'il)—*the father is strength.*
Nu. 3:35. prince was Zuriel son of A.
1 Ch. 2:29. wife of Abishur was A.
5:14. These were the sons of A.
2 Ch. 11:18. Rehob. took wife of dau. of A.
Es. 2:15. Esther, dau. of A. (9:29)

ABIHU (A-bi'hu)—*my father is He, i.e. God.*
Son of Aaron who was destroyed for offering strange fire before Jehovah.
Ex. 6:23. she bare him Nadab and A.
24: 1. Come up unto JH. thou and A.
9. then went up Moses, & A. etc.
28: 1. may minister, even Aaron, A. etc.
Le. 10: 1. Nadab and A. took his censer
Nu. 3: 2. Nadab the first-born, and A.
4. Nadab & A. died bef. JH. (26:61)

Indeksy

Tablice sufiksów

Udi Manber

Izraelski informatyk. W 1990 roku wraz z Genem Myersem wprowadził pojęcie tablicy sufiksów. Był wiceprezesem w Amazonie, Google i YouTube.

Gene Myers (31.12.1953–)

Amerykański informatyk i bioinformatyk. Jest współautorem programu BLAST (basic local alignment search tool), który porównuje łańcuchy w biologicznych bazach danych. Artykuł z 1990 roku, w którym opisano BLAST, był do 2024 roku cytowany ponad 180 tysięcy razy.

Tablice sufiksów

Tablica sufiksów zawiera posortowane indeksy tych pozycji w danym łańcuchu T, od których zaczynają się sufiksy tego łańcucha. Tak sortuję elementy tablicy sufiksów:

Jeśli pewien sufiks S_1 poprzedza inny sufiks S_2 w porządku leksykograficznym, to indeks tej pozycji łańcucha T, od której zaczyna się sufiks S_1, znajduje się w tablicy sufiksów łańcucha T przed indeksem tej pozycji łańcucha T, od której zaczyna się sufiks S_2

Tablice sufiksów – przykład

Tablica sufiksów łańcucha ananas to `[0 2 4 1 3 5]`

od pozycji `0` łańcucha ananas zaczyna się jego sufiks ananas

od pozycji `2` łańcucha ananas zaczyna się jego sufiks anas

od pozycji `4` łańcucha ananas zaczyna się jego sufiks as

od pozycji `1` łańcucha ananas zaczyna się jego sufiks nanas

od pozycji `3` łańcucha ananas zaczyna się jego sufiks nas

od pozycji `5` łańcucha ananas zaczyna się jego sufiks s

Tablice sufiksów – najprostszy algorytm tworzenia

```
type suffix struct {
    s []byte
    n int
}

func sortedSuffixes(text []byte) []suffix {
    suffixes := make([]suffix, len(text))
    for i := range suffixes {
        suffixes[i].s = text[i:]
        suffixes[i].n = i
    }
    sort.Slice(suffixes, func(i, j int) bool {
        return bytes.Compare(suffixes[i].s, suffixes[j].s) < 0
    })
    return suffixes
}
```

Tablice sufiksów – najprostszy algorytm tworzenia

```
// Index implementuje indeks łańcucha `text`. Ten indeks
// składa się z tablicy sufiksów łańcucha `text`
// i z łańcucha `text`
type Index struct {
    suffixes []int
    text     []byte
}
```

Tablice sufiksów – najprostszy algorytm tworzenia

```
// SuffixArray zwraca indeks łańcucha `text`
func SuffixArray(text []byte) Index {
    index := Index{
        make([]int, len(text)),
        make([]byte, len(text)),
    }
    index.text = text
    for i, suf := sortedSuffixes(text) {
        index.suffixes[i] = suf.n
    }
    return index
}
```

Tablice sufiksów – najprostszy algorytm tworzenia

Za pomocą funkcji `SuffixArray` mogę stworzyć tablicę sufiksów łańcucha `text` w czasie $O(|\texttt{text}|^2 \log|\texttt{text}|)$

Istnieją szybsze algorytmy tworzenia tablic sufiksów. Najszybsze z tych algorytmów działają w czasie $O(|\texttt{text}|)$

Tablice sufiksów – zajmowana pamięć

```
// Index implementuje indeks łańcucha `text`. Ten indeks
// składa się z tablicy sufiksów łańcucha `text`
// i z łańcucha `text`
type Index struct {
    suffixes []int
    text     []byte
}
```

Wewnątrz struktury `Index`:

- każdy element tablicy `suffixes` zajmuje 1 słowo maszynowe
- każdy znak łańcucha `text` zajmuje 1 bajt

Zatem tablica sufiksów łańcucha `text` zajmuje `(sizeof(int)+1)*len(text)` bajtów

Zastosowania tablic sufiksów

Zastosowania tablic sufiksów – 1

Tak wyszukuję wszystkie wystąpienia danego wzorca P w danym tekście T:

- Buduję tablicę sufiksów `suffixes` łańcucha T
- Znajduję indeks `i` pierwszego takiego elementu tablicy `suffixes`, który nie poprzedza wzorca T w porządku leksykograficznym
- Znajduję indeks `j` pierwszego takiego elementu tablicy `suffixes`, którego prefiks o długości $|P|$ występuje w porządku leksykograficznym za wzorcem P
- Wycinek `suffixes[i:j]` zawiera indeksy tych pozycji, od których zaczynają się wystąpienia wzorca P w tekście T

Zastosowania tablic sufiksów – 1

```
// Suffix zwraca ten sufiks łańcucha `x.text`, który
// zaczyna się od `i`-tej pozycji tego łańcucha
func (x Index) Suffix(i int) []byte {
    return x.text[i:]
}
```

Zastosowania tablic sufiksów – 1

```
// LookupAll zwraca wycinek, którego elementami są wszystkie
// te pozycje, na których łańcuch `s` występuje w łańcuchu `x.text`
func (x Index) LookupAll(s []byte) []int {
    i := sort.Search(len(x.suffixes), func(i int) bool {
        return bytes.Compare(x.Suffix(x.suffixes[i]), s) >= 0
    })
    j := i + sort.Search(len(x.suffixes) - i, func(j int) bool {
        return !bytes.HasPrefix(x.Suffix(x.suffixes[i+j]), s)
    })
    return x.suffixes[i:j]
}
```

Zastosowania tablic sufiksów – 1

Tak wyszukuję wszystkie wystąpienia wzorca `issi` w tekście `mississippi`:

```
10: i
 7: ippi
>4: issippi
>1: ississippi
 0: mississippi
 9: pi
 8: ppi
 6: sippi
 3: sissippi
 5: ssippi
 2: ssissippi
```

Wzorzec `issi` występuje w tekście `mississippi` na pozycjach 4 i 1

Zastosowania tablic sufiksów – 1

Korzystając z tablicy sufiksów, mogę znaleźć wszystkie wystąpienia danego wzorca P w danym tekście T w czasie $O(\log|T|)$ dzięki temu, że w funkcji `sort.Search` zastosowano wyszukiwanie binarne, które działa w czasie $O(\log|T|)$

Zastosowania tablic sufiksów – 2

```
// lenOfCommonPrefix zwraca długość najdłuższego
// wspólnego prefiksu łańcuchów `s` i `t`
func lenOfCommonPrefix(s, t []byte) int {
    k := 0
    for ; k < min(len(s), len(t)); k++ {
        if s[k] != t[k] {
            return k
        }
    }
    return k
}
```

Zastosowania tablic sufiksów – 2

```
// LongestRepeatingSubstring znajduje najdłuższy taki łańcuch,
// który występuje w łańcuchu `text` co~najmniej `k` razy
func LongestRepeatingSubstring(text []byte, k int) []byte {
    index := SuffixArray(text)
    substr := []byte{}
    for i := 0; i + k <= len(a); i++ {
        m := lenOfCommonPrefix(
            index.Suffix(i), index.Suffix(i+k-1))
        if m > len(substr) {
            substr = index.Suffix(i)[:m]
        }
    }
    return substr
}
```

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`i`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`i`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Tak wyszukuję najdłuższy taki łańcuch, który powtarza się 2 razy w tekście `mississippi`:

```
 10: i
  7: ippi
  4: issippi
  1: ississippi
  0: mississippi
  9: pi
  8: ppi
  6: sippi
  3: sissippi
  5: ssippi
  2: ssissippi
```

`issi`

Zastosowania tablic sufiksów – 2

Korzystając z tablic sufiksów mogę znaleźć najdłuższy taki łańcuch, który występuje co najmniej k razy w łańcuchu T, w czasie $O(|T|^2)$

Drzewa sufiksów

Peter Weiner

Amerykański informatyk i przedsiębiorca. Był promotorem doktoratu Briana Kernighana. Założył Interactive Systems Corporation, pierwszą komercyjną firmę zajmującą się UNIX-em. Wprowadził pojęcie drzewa sufiksów. Jest autorem algorytmu tworzenia drzew sufiksów, który działa w czasie liniowym względem długości łańcucha. Donald Knuth nazwał ten algorytm „algorytmem roku 1973”.

Drzewa trie

Drzewo trie – takie drzewo wyszukiwania, którego krawędzie są opisane przez fragmenty kluczy wyszukiwania

Nazwa trie pochodzi od wyrazu retrieval

Nieskompresowane drzewo trie – takie drzewo trie, którego krawędzie są opisane przez pojedyncze znaki

Skompresowane drzewo trie – takie drzewo trie, w którym każdy taki węzeł, z którego wychodzi dokładnie 1 krawędź, został połączony ze swoim rodzicem w 1 węzeł. Krawędzie skompresowanego drzewa trie są opisane przez niepuste podłańcuchy kluczy wyszukiwania

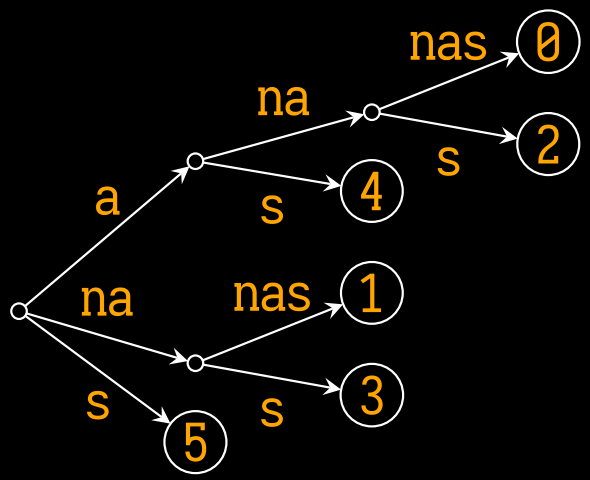
Drzewa sufiksów

Drzewo sufiksów łańcucha T to takie skompresowane drzewo trie, w którym:

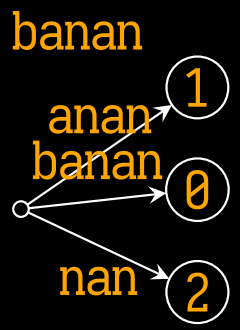
- Każda krawędź między węzłami jest opisana przez pewien niepusty podłańcuch łańcucha T
- Z każdego węzła wewnętrznego wychodzą 2 krawędzie lub więcej krawędzi
- Z żadnego węzła nie wychodzą takie 2 krawędzie, których opis zaczynałby się tym samym znakiem
- Jest $|T|$ liści. Te liście mają numery od 0 do $T - 1$
- Gdy połączę etykiety każdej krawędzi na ścieżce od korzenia drzewa do i-tego liścia tego drzewa, otrzymam sufiks $T[i:]$

Drzewa sufiksów – przykład

ananas



Drzewa sufiksów



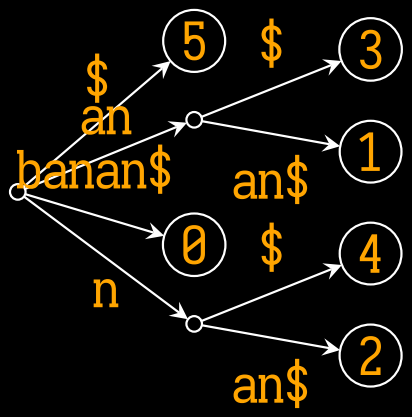
Jeśli pewien sufiks łańcucha T jest prefiksem innego sufiksu łańcucha T, czyli jeśli pewien sufiks łańcucha T jest równy innemu podłańcuchowi łańcucha T, to drzewo sufiksów łańcucha T ma mniej liści niż powinno mieć

Na przykład w drzewie sufiksów łańcucha banan brakuje liści 3 i 4 na końcach krawędzi an i n

W takim przypadku budujemy drzewa sufiksów takich łańcuchów, na których końcu jest dodany wartownik. Ten wartownik jest zwykle oznaczany znakiem dolara $

Drzewa sufiksów – przykład

banan$



Drzewo sufiksów – implementacja

Jak przechowywać węzły drzewa sufiksów? Jest kilka możliwości:

- Tablica wskaźników do węzłów potomnych, która ma tyle elementów, ile alfabet ma znaków
- Lista wskaźników do węzłów potomnych, posortowana według znaków
- Drzewo, na przykład drzewo binarne. Etykietami krawędzi tego drzewa są znaki, a węzły tego drzewa zawierają wskaźniki do węzłów potomnych
- Tablica haszująca znaki na wskaźniki do węzłów potomnych

Drzewo sufiksów – zajmowana pamięć

Przykładowa implementacja krawędzi drzewa sufiksów i węzła drzewa sufiksów:

```
type edge struct {
    firstChar  byte
    targetNode int
}
type node struct {
    start, end int
    suffixLink int
    outEdges   []edge
}
```

Drzewo sufiksów – zajmowana pamięć

- Krawędź drzewa sufiksów:
  - 2 słowa maszynowe – pierwszy znak etykiety i węzeł docelowy
- Węzeł drzewa sufiksów:
  - 2 słowa maszynowe – indeksy początku i końca podłańcucha, który opisuje krawędź
  - 1 słowo maszynowe – łącze do sufiksu, o którym będzie mowa później
  - 3 słowa maszynowe na początek, długość i pojemność wycinka `outEdges`
- Łańcuch `text`:
  - 1 bajt na każdy znak

Drzewo sufiksów – zajmowana pamięć

Drzewo sufiksów łańcucha `text` ma:

- `len(text)` liści
- co najwyżej `len(text)` węzłów wewnętrznych
- o 1 krawędź mniej niż ma węzłów

Zatem podana implementacja drzewa sufiksów łańcucha `text` zajmuje:

- co najwyżej `(2*len(text))*6` słów maszynowych na węzły
- co najwyżej `(2*len(text))*2` słowa maszynowe na krawędzie
- `len(text)` bajtów na łańcuch `text`

Zatem drzewo sufiksów zajmuje co najwyżej
`(16*sizeof(int)+1)*len(text)` bajtów

Drzewa sufiksów – wyzwanie 1

Udowodnię, że drzewo sufiksów łańcucha T:

- ma tyle samo liści, ile łańcuch T ma znaków
- ma mniej wierzchołków wewnętrznych niż łańcuch T ma znaków,

czyli że drzewo sufiksów łańcucha T ma rozmiar $O(|T|)$

Drzewa sufiksów – wyzwanie 2

Spośród łańcuchów, które mają 5 znaków, podam łańcuch, którego drzewo sufiksów ma najwięcej krawędzi

Zastosowania drzew sufiksów

Zastosowania drzew sufiksów – 1

Tak sprawdzam, czy wzorzec P występuje w tekście T:

- Buduję drzewo sufiksów łańcucha T
- Szukam w drzewie sufiksów łańcucha T takiej ścieżki od korzenia, że połączone etykiety krawędzi, które składają się na tę ścieżkę, tworzą łańcuch P
- Jeśli w drzewie sufiksów istnieje taka ścieżka, to znaczy, że wzorzec P występuje w tekście T
- Jeśli w drzewie sufiksów nie ma takiej ścieżki, to znaczy, że wzorzec P nie występuje w tekście T

Zastosowania drzew sufiksów – 2

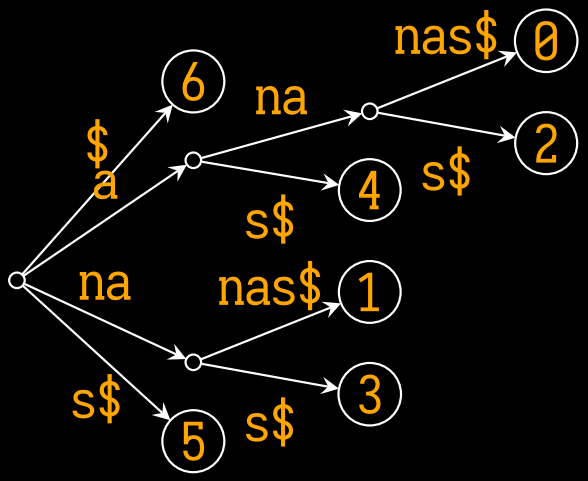
Znam tekst T. Ten tekst się nie zmienia. Chcę znaleźć wszystkie wystąpienia każdego z wielu wzorców $P_1, P_2, \ldots P_n$ w tym tekście

Tworzę drzewo sufiksów łańcucha T

Przechodzę ścieżką od korzenia drzewa wzdłuż tych krawędzi, których etykiety łączą się we wzorzec P_i. W ten sposób dochodzę do pewnego węzła N. Potem przeszukuję węzły potomne węzła N, na przykład w głąb. Każdy liść drzewa, który znajduję, zawiera indeks pozycji, od której zaczyna się pewne wystąpienie wzorca P_i w tekście

W ten sposób mogę znaleźć w tekście T wszystkie wystąpienia dowolnego wzorca P_i w czasie $O(|P_i| + \text{liczba wystąpień wzorca } P_i \text{ w tekście } T)$. Ten czas nie zależy od długości łańcucha T

Zastosowania drzew sufiksów – 2



Wystąpienia wzorca na zaczynają się od pozycji 1 i 3

Wystąpienia wzorca an zaczynają się od pozycji 0 i 2

Wystąpienia wzorca a zaczynają się od pozycji 0, 2 i 4

Wystąpień wzorca x nie ma

Zastosowania drzew sufiksów – 3

Jak znaleźć wszystkie te łańcuchy ze zbioru $\{T_1, T_2, \ldots T_n\}$, które zawierają dany wzorzec P?

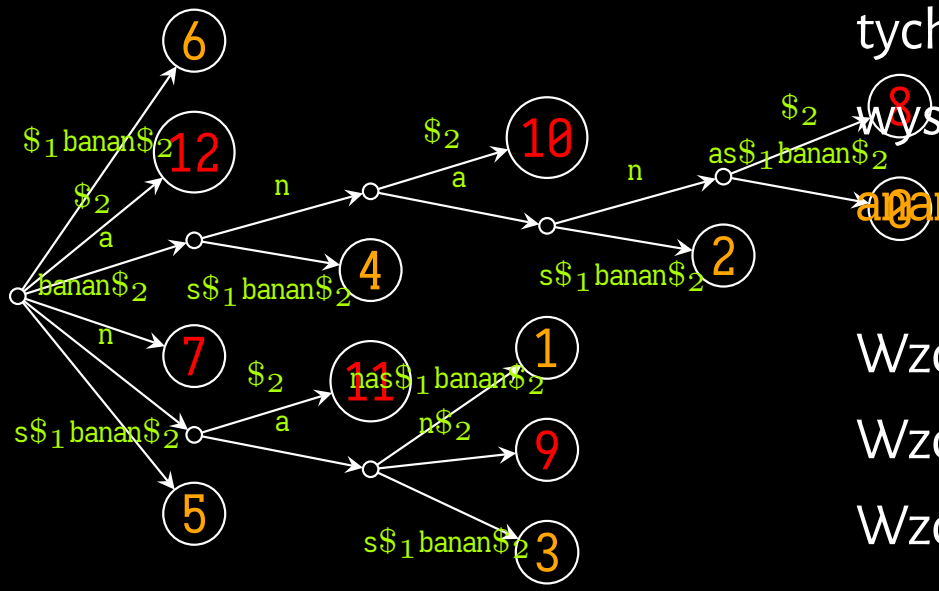
Tworzę drzewo sufiksów łańcucha $T_1\$_1 T_2\$_2 \ldots T_n\$_n$

Przechodzę ścieżką od korzenia drzewa sufiksów łańcucha $T_1\$_1 T_2\$_2 \ldots T_n\$_n$ wzdłuż tych krawędzi, których etykiety łączą się we wzorzec P. W ten sposób dochodzę do pewnego węzła N

Potem przeszukuję węzły potomne węzła N, na przykład w głąb. Każdy liść drzewa, który znajduję, zawiera indeks pozycji, od której zaczyna się pewne wystąpienie wzorca P w tekście

Zastosowania drzew sufiksów – 3

Jak znaleźć wszystkie te łańcuchy ze zbioru $\{T_1, T_2, \ldots T_n\}$, które zawierają dany wzorzec P?



Wyznaczam te łańcuchy, które zawierają dany wzorzec, na podstawie indeksów tych pozycji, od których zaczynają się wystąpienia tego wzorca w tekście ananas$1 banan$2

Wzorzec an występuje w łańcuchach 1 i 2

Wzorzec na występuje w łańcuchach 1 i 2

Wzorzec as występuje w łańcuchu 1

Wzorzec ban występuje w łańcuchu 2

Wzorzec x nie występuje w żadnym z tych łańcuchów

Zastosowania drzew sufiksów – 3

Jak znaleźć wszystkie łańcuchy ze zbioru $\{T_1, T_2, \ldots T_n\}$, które zawierają dany wzorzec P?

Przykład zastosowania: Baza danych DNA mitochondrialnego żołnierzy USA

Pobieram DNA mitochondrialne od żyjących żołnierzy. Sekwencjonuję mały fragment DNA każdego żołnierza. Wybieram ten fragment tak, żeby był różny u różnych żołnierzy. Zapisuję ten fragment w bazie danych jako identyfikator żołnierza

Tak identyfikuję poległych żołnierzy:

Pobieram DNA mitochondrialne z ciał poległych żołnierzy. Sekwencjonuję ten sam fragment DNA. Porównuję ten fragment z identyfikatorami żołnierzy. Może się zdarzyć, że z ciała nie da się pobrać pełnego fragmentu DNA. Dlatego szukam takich identyfikatorów, które zawierają pobrany fragment

Zastosowania drzew sufiksów – 4

Jak znaleźć najdłuższy wspólny podłańcuch 2 łańcuchów T_1 i T_2?

Buduję drzewo sufiksów łańcucha $T_1\$_1T_2\$_2$. Dodaję do każdego z węzłów wewnętrznych tego drzewa 2 bity

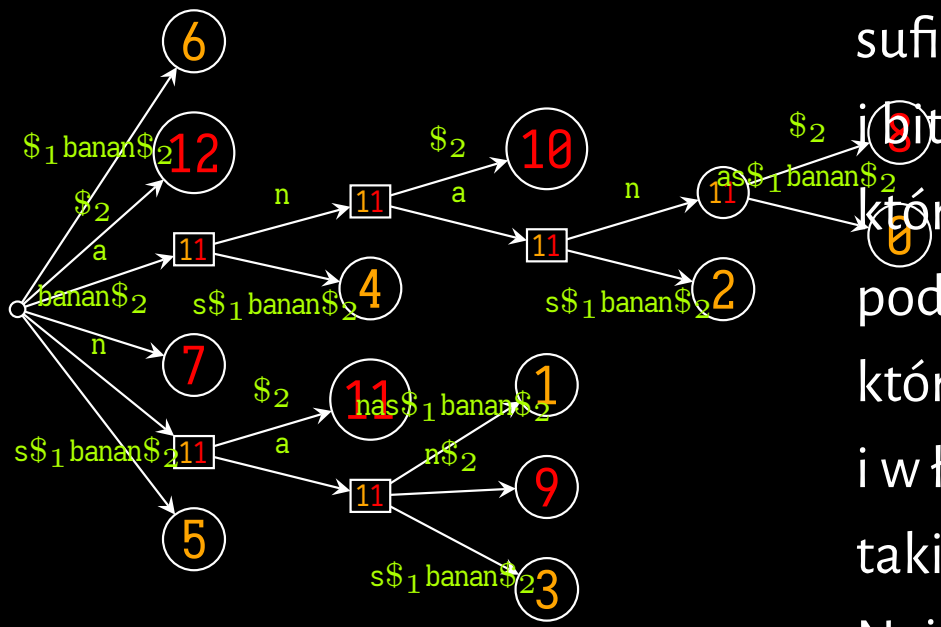
W każdym węźle wewnętrznym N tego drzewa ustawiam odpowiednie bity:

- bit 0, jeśli dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_1\$_1T_2\$_2$
- bit 1, jeśli dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_2\$_2$
- bit 0 i bit 1, jeśli dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_1\$_1T_2\$_2$ i dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_2\$_2$

Obliczam 2 bity dodane do każdego węzła wewnętrznego jako sumę logiczną (OR) 2 bitów dodanych do dzieci tego węzła, przeszukując drzewo sufiksów w głąb w kolejności wstecznej (post-order), czyli odwiedzając najpierw dzieci każdego węzła wewnętrznego, a potem ten węzeł

Zastosowania drzew sufiksów – 4

Jak znaleźć najdłuższy wspólny podłańcuch łańcuchów ananas i banan?



Przeszukuję od korzenia te węzły drzewa sufiksów, w których jest ustawiony i bit 0, i bit 1. Etykiety tych krawędzi, wzdłuż których przechodzę, łączą się w takie podłańcuchy łańcucha ananas$\$_1$banan$\$_2$, które występują i w łańcuchu ananas, i w łańcuchu banan. Szukam najdłuższego takiego wspólnego podłańcucha Najdłuższy wspólny podłańcuch łańcuchów ananas i banan to łańcuch ana

Zastosowania drzew sufiksów – 5

Jak wykrywać zanieczyszczenia w DNA?

Przykład: DNA rzekomo wyekstrahowane z kości dinozaura okazało się bardziej podobne do ludzkiego DNA niż do DNA ptaków i krokodyli

Dany jest łańcuch DNA T_1, połączone łańcuchy DNA możliwych zanieczyszczeń T_2 i liczba całkowita k. Chcę znaleźć wszystkie takie podłańcuchy łańcucha T_2, które występują w łańcuchu T_1 i mają długość większą niż k

Zastosowania drzew sufiksów – 5

Jak wykrywać zanieczyszczenia w DNA?

Buduję drzewo sufiksów łańcucha $T_1\$_1T_2\$_2$. Dodaję do każdego z węzłów wewnętrznych tego drzewa 2 bity

W każdym węźle wewnętrznym N tego drzewa ustawiam odpowiednie bity na 1:

- bit 0, jeśli dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_1\$_1T_2\$_2$
- bit 1, jeśli dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_2\$_2$
- bit 0 i bit 1, jeśli dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_1\$_1T_2\$_2$ i dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_2\$_2$

Zastosowania drzew sufiksów – 5

Wyznaczam 2 bity dodane do każdego węzła wewnętrznego jako sumę logiczną (OR) 2 bitów dodanych do dzieci tego węzła, przeszukując drzewo sufiksów w głąb w kolejności wstecznej (post-order), czyli odwiedzając najpierw dzieci każdego węzła wewnętrznego, a potem ten węzeł

Przeszukuję od korzenia te węzły drzewa sufiksów, w których jest ustawiony i bit 0, i bit 1. Etykiety tych krawędzi, wzdłuż których przechodzę, łączą się w takie podłańcuchy łańcucha $T_1\$_1T_2\$_2$, które występują i w łańcuchu T_1, i w łańcuchu T_2. Szukam wszystkich takich podłańcuchów, których długość jest większa niż dana stała k

Zastosowania drzew sufiksów – 6

Ile razy dany wzorzec P występuje w danym tekście T?

Buduję drzewo sufiksów łańcucha T

Dodaję licznik do każdego węzła tego drzewa. W tym liczniku zapamiętuję, ile liści ma poddrzewo, którego korzeniem jest ten węzeł. Wyznaczam wartość licznika dodanego do każdego węzła jako sumę wartości liczników dodanych do dzieci tego węzła, przeszukując drzewo sufiksów w głąb w kolejności wstecznej (post-order), czyli odwiedzając najpierw dzieci każdego węzła wewnętrznego, a potem ten węzeł

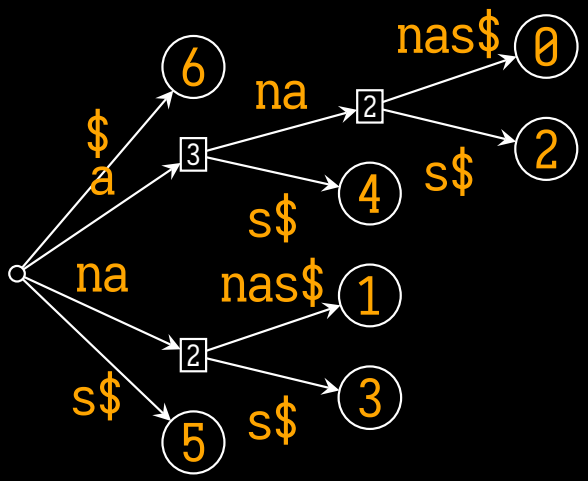
Zastosowania drzew sufiksów – 6

Ile razy dany wzorzec P występuje w danym tekście T?

Przechodzę ścieżką od korzenia tego drzewa sufiksów wzdłuż tych krawędzi, których etykiety składają się we wzorzec P. W ten sposób dochodzę do pewnego węzła N. Odczytuję z licznika węzła N liczbę liści w poddrzewie węzła N. Gdy nie ma takiego węzła, to znaczy, że wzorzec P występuje 0 razy w tekście T

Zastosowania drzew sufiksów – 6

Ile razy dany wzorzec występuje w danym tekście?



Zastosowania drzew sufiksów – 7

Jak znaleźć najdłuższy podłańcuch wspólny dla wielu różnych łańcuchów T_1, T_2, ..., T_n, ale niekoniecznie wspólny dla wszystkich tych łańcuchów?

Przykład: DNA wielu różnych gatunków. Mutacje tych fragmentów DNA, które kodują krytyczne funkcje, i mutacje innych fragmentów DNA zdarzają się z jednakowym prawdopodobieństwem. Jednak mutacje tych fragmentów DNA, które kodują krytyczne funkcje, częściej powodują przedwczesną śmierć organizmu, przez co ten organizm nie ma potomków. Zatem te fragmenty DNA, które kodują krytyczne funkcje, mniej się zmieniają z pokolenia na pokolenie.

Gdy znajdę najdłuższe wspólne podłańcuchy DNA wielu gatunków, dowiem się, które fragmenty DNA kodują krytyczne funkcje

Zastosowania drzew sufiksów – 7

Jak znaleźć najdłuższy podłańcuch wspólny dla wielu różnych łańcuchów T_1, T_2, ..., T_n, ale niekoniecznie wspólny dla wszystkich tych łańcuchów?

Buduję drzewo sufiksów łańcucha $T_1\$_1T_2\$_2 \ldots T_n\$_n$. Przypisuję do każdego z węzłów wewnętrznych tego drzewa tablicę n bitów. k-ty bit tej tablicy w węźle wewnętrznym N jest ustawiony na 1, gdy dowolny liść tego poddrzewa, którego korzeniem jest węzeł N, odpowiada pewnemu sufiksowi łańcucha $T_1\$_1 \ldots T_k\$_k \ldots T_n\$_n$, który zaczyna się w łańcuchu T_k

Zastosowania drzew sufiksów – 7

Obliczam tablicę bitów każdego węzła wewnętrznego jako sumę logiczną (OR) tablic bitów dodanych do dzieci tego węzła, przeszukując drzewo sufiksów w głąb w kolejności wstecznej (post-order), czyli odwiedzając najpierw dzieci każdego węzła wewnętrznego, a potem ten węzeł

W tablicy bitów węzła N tyle bitów jest ustawionych na 1, ile różnych numerów łańcuchów pojawia się w liściach tego poddrzewa, którego korzeniem jest węzeł N

Przy ustalonej liczbie ℓ przeszukuję od korzenia takie węzły drzewa sufiksów, że w tablicach bitów tych węzłów co najmniej ℓ bitów jest ustawionych. Etykiety tych krawędzi, wzdłuż których przechodzę, łączą się w podłańcuchy łańcucha $T_1\$_1T_2\$_2 \ldots T_n\$_n$. Szukam najdłuższego takiego podłańcucha

Algorytm Ukkonena

Tę część wykładu opracowałem na podstawie odpowiedzi Johannesa Gollera ze Stack Overflow: `https://stackoverflow.com/questions/9452701/ukkonens-suffix-tree-algorithm-in-plain-english`

Esko Ukkonen (26.1.1950–)

Fiński informatyk. Emerytowany profesor Uniwersytetu Helsińskiego.
W 1995 roku opublikował jeden z prostszych algorytmów budowania drzew sufiksów. Ten algorytm działa w czasie liniowym względem długości łańcucha. W 2000 roku Esko Ukkonen otrzymał najwyższe odznaczenie Finlandii, Order Białej Róży Finlandii.

Algorytm Ukkonena

Drzewo sufiksów to skompresowane drzewo trie. Etykiety jego krawędzi to pary liczb całkowitych [od,do]. Liczby od i do to indeksy początku podłańcucha i końca podłańcucha. Chociaż etykieta każdej krawędzi może mieć dowolną długość, zajmuje tylko $O(1)$ komórek pamięci

Algorytm Ukkonena wykonuje kroki, przeglądając znaki łańcucha T od początku do końca tego łańcucha. Na każdy znak ciągu przypada jeden krok. Chociaż każdy krok może obejmować więcej niż jedną pojedynczą operację, wykażę, że całkowita liczba operacji wynosi $O(|T|)$, i że każda z tych operacji zajmuje czas $O(1)$

Algorytm Ukkonena

abcabxabcd

Krok 1: Zaczynam od lewej. Najpierw dodaję do drzewa sufiksów tylko pojedynczy znak a, tworząc krawędź od korzenia drzewa, czyli od węzła `root`, do liścia i oznaczając ją [0,#]. Oznacza to, że krawędź reprezentuje podłańcuch, który zaczyna się na pozycji 0 i kończący na bieżącym końcu. Symbolem # oznaczam bieżący koniec, który znajduje się na pozycji 1 (zaraz za a)

Zatem początkowe drzewo sufiksów wygląda tak:

Algorytm Ukkonena

abcabxabcd

Krok 2: Teraz przechodzę do pozycji 2 (zaraz za b). Celem każdego kroku jest dodać wszystkie sufiksy aż do bieżącej pozycji. Aby osiągnąć ten cel:

- rozszerzam istniejącą krawędź a do ab
- wstawiam jedną nową krawędź dla b

Teraz drzewo sufiksów wygląda tak:

czyli

Algorytm Ukkonena

:

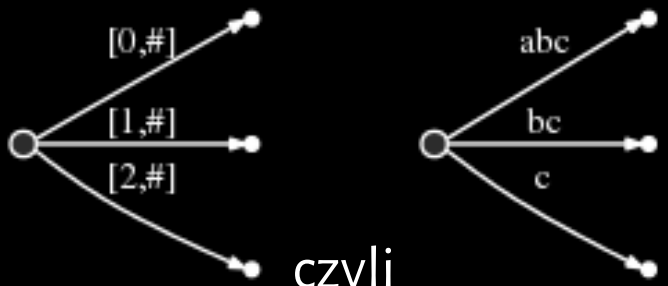
- Reprezentacja krawędzi ab jest taka sama jak w początkowym drzewie: [0,#]. Jej znaczenie samo się zmieniło, bo zmieniłem bieżącą pozycję # z 1 na 2
- Każda krawędź zajmuje $O(1)$ miejsca, bo składa się tylko z dwóch wskaźników do tekstu, niezależnie od tego, jak długiemu podłańcuchowi odpowiada

Algorytm Ukkonena

abcabxabcd

Krok 3: Znowu zwiększam pozycję o 1 i aktualizuję drzewo sufiksów, dodając znak c na końcu każdej istniejącej krawędzi i wstawiając jedną nową krawędź dla nowego sufiksu c

Teraz drzewo sufiksów wygląda tak:



czyli

Algorytm Ukkonena

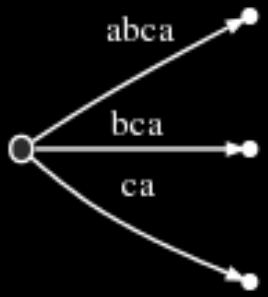
Zauważam, że:

- Po każdym kroku drzewo to trie, które buduję, jest poprawnym drzewem sufiksów prefiksu łańcucha T aż do bieżącej pozycji
- Wykonałem tyle kroków, ile przeczytałem znaków
- W każdym kroku wykonałem $O(1)$ operacji, bo zwiększanie zmiennej # o 1 zajmuje czas $O(1)$ i jedną nową krawędź dla ostatniego znaku można wstawić w czasie $O(1)$
- Zatem drzewo sufiksów takiego łańcucha `text`, w którym znaki się nie powtarzają, można zbudować w czasie $O(|\texttt{text}|)$

Algorytm Ukkonena

abcabxabcd

Krok 4: Przesuwam # na pozycję 4. W ten sposób niejawnie przesuwam końce wszystkich krawędzi aż do tej pozycji. Teraz drzewo sufiksów wygląda tak:



Algorytm Ukkonena

Teraz należy wstawić do korzenia a, ostatni sufiks bieżącego kroku

Zanim to zrobię, wprowadzam oprócz zmiennej # jeszcze 2 zmienne:

- Aktywne położenie: trójka (`active_node`, `active_edge`, `active_length`)
- `backlog`: liczba całkowita, która wskazuje, ile nowych sufiksów należy dodać do drzewa

Algorytm Ukkonena

- Kiedy wstawiałem do drzewa sufiksów znaki `abc`, aktywnym położeniem było zawsze (`root`, `'$'`, `0`), czyli aktywnym węzłem był korzeń drzewa, aktywną krawędzią był znak wartownika, a aktywna długość była równa 0. Zatem każda nowa krawędź, którą dodawałem do drzewa sufiksów, była dodawana do korzenia drzewa
- Na początku każdego kroku zmienna `backlog` była ustawiana na 1. Oznaczało to, że na końcu każdego kroku należało wstawić do drzewa sufiksów tylko 1 sufiks

Algorytm Ukkonena

Teraz znak a znowu występuje w łańcuchu. Dlatego aktywne położenie i zmienna `backlog` będą się zmieniać. Kiedy wstawiam do korzenia drzewa bieżący ostatni znak a, zauważam, że z korzenia już wychodzi krawędź, która zaczyna się od znaku a: krawędź abca. W takim przypadku:

- Nie dodaję nowej krawędzi [4,#] do korzenia drzewa. Zamiast tego zauważam, że sufiks a już jest dodany do drzewa. Nie przeszkadza mi to, że ten sufiks kończy się wewnątrz dłuższej krawędzi.
- Ustawiam aktywne położenie na (`root`, `'a'`, `1`). Oznacza to, że aktywne położenie znajduje się za pozycją 1 tej krawędzi, która wychodzi z węzła `root` i ma etykietę a. Zauważam, że pierwszy znak krawędzi jednoznacznie wyznacza tę krawędź, bo z danego węzła może wychodzić tylko jedna krawędź, która zaczyna się od danego znaku
- Zwiększam wartość zmiennej `backlog` o 1. Teraz ta zmienna ma wartość 2

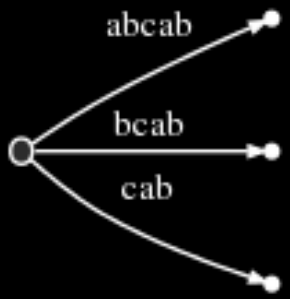
Algorytm Ukkonena

Zauważam, że gdy ostatni sufiks, który należy wstawić do drzewa sufiksów, już występuje w drzewie sufiksów, nie zmieniam drzewa sufiksów. W takim przypadku zmieniam tylko bieżące położenie i wartość zmiennej `backlog`. Mogę je zmienić w czasie $O(1)$

Algorytm Ukkonena

abcabxabcd

Krok 5: Zmieniam bieżącą pozycję # na 5. Teraz drzewo sufiksów wygląda tak:



Algorytm Ukkonena

Zmienna `backlog` ma wartość 2, więc należy wstawić do drzewa sufiksów 2 ostatnie sufiksy tego podłańcucha, który kończy się na bieżącej pozycji

- Sufiks `a` z poprzedniego kroku nie został wstawiony do drzewa sufiksów. W bieżącym kroku zwiększyłem bieżącą pozycję o 1, więc ten sufiks urósł z `a` do `ab`
- Należy też wstawić do drzewa sufiksów nowy ostatni sufiks, czyli `b`

Algorytm Ukkonena

Chcę wstawiać te 2 sufiksy do drzewa sufiksów w kolejności od najdłuższego do najkrótszego sufiksu

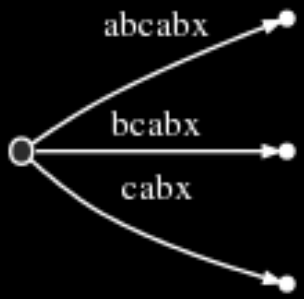
Aktywne położenie wskazuje, gdzie kończy się podłańcuch `a` i należy wstawić sufiks `b`: za pierwszym znakiem `a` na krawędzi `abcab`. Okazuje się, że znak `b` już jest na krawędzi `abcab` za pierwszym znakiem `a`. Zatem nie zmieniam drzewa sufiksów, tylko:

- Zmieniam aktywne położenie na (`root`, `'a'`, `2`). Oznacza to, że aktywne położenie znajduje się na tej samej krawędzi `abcab` wychodzącej z tego samego węzła `root`, o 1 znak dalej, czyli za pierwszym znakiem `b`
- Zwiększam wartość zmiennej `backlog` do 3

Algorytm Ukkonena

abcabxabcd

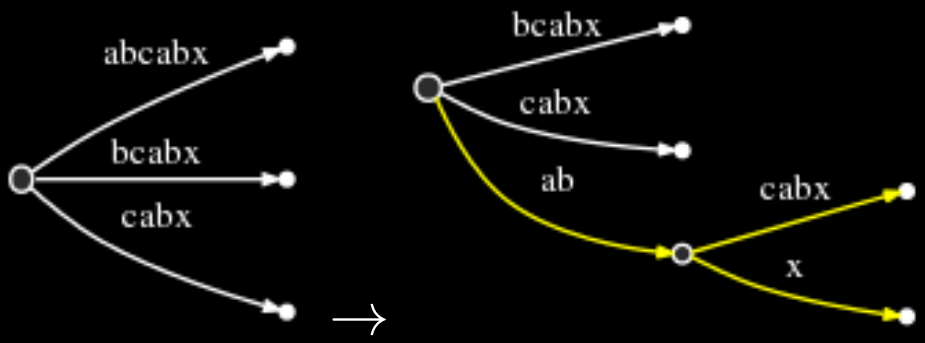
Krok 6: Zwiększam wartość zmiennej # o 1. Teraz drzewo sufiksów wygląda tak:



Algorytm Ukkonena

`abcabxabcd`

Zmienna `backlog` ma wartość 3. Zatem należy wstawić do drzewa sufiksów sufiksy `abx`, `bx` i `x`. Aktywne położenie wskazuje, gdzie kończy się podłańcuch `ab` i należy wstawić sufiks `x`: za pierwszym znakiem `b` krawędzi `abcabx` wychodzącej z węzła `root`. Znaku `x` nie ma w tym miejscu krawędzi `abcabx`. Zatem dzielę krawędź `abcabx` i wstawiam do drzewa sufiksów nowy węzeł wewnętrzny:



Algorytm Ukkonena

Każda krawędź zawiera indeksy początku i końca podłańcucha, zatem mogę podzielić krawędź i wstawić do drzewa sufiksów nowy węzeł wewnętrzny w czasie $O(1)$

Poradziłem sobie z sufiksem `abx`. Zmniejszam wartość zmiennej `backlog` do 2. Chcę wstawić do drzewa sufiksów następny zaległy sufiks, czyli `bx`. Najpierw jednak należy zaktualizować aktywne położenie. Korzystam z tej reguły:

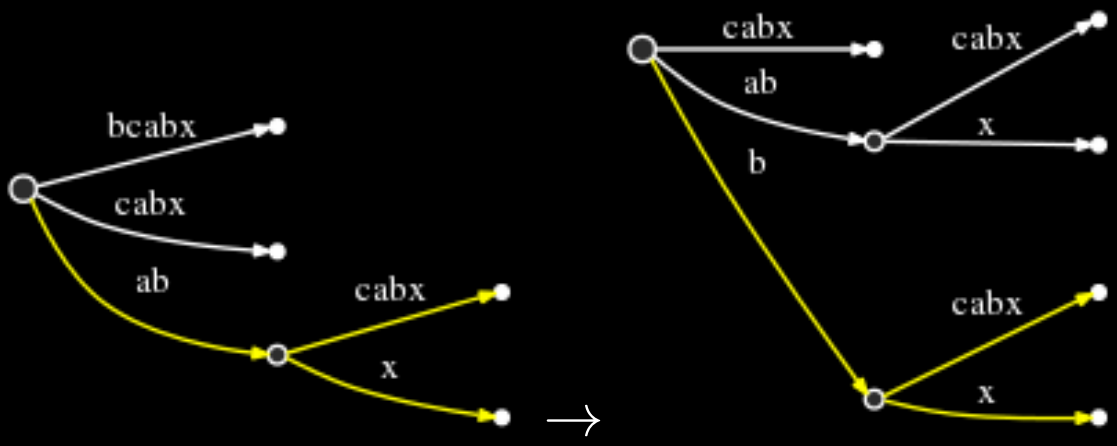
`Reguła 1:` Jeśli `active_node == root` i `active_length > 0`, to:

- `active_node` nadal będzie mieć wartość `root`
- Przesuwam `active_edge` w prawo do pierwszego znaku następnego zaległego sufiksu
- Zmniejszam `active_length` o 1

Algorytm Ukkonena

abcabxabcd

Zatem nowe aktywne położenie (`root` , `'b'` ,1) wskazuje, że następny sufiks należy wstawić na krawędzi `bcabx`, za pierwszym znakiem, czyli za znakiem `b`. Mogę znaleźć to miejsce w czasie $O(1)$ i sprawdzić, czy znak `x` już tam jest, czy nie. Gdyby znak `x` tam był, zakończyłbym krok 6. Ale znaku `x` tam nie ma, więc wstawiam go, dzieląc krawędź `bcabx`:



Algorytm Ukkonena

W czasie $O(1)$ podzieliłem krawędź i zmieniłem:

- wartość zmiennej `backlog` na 1
- aktywne położenie na (`root`, `'x'`, `0`) zgodnie z Regułą 1

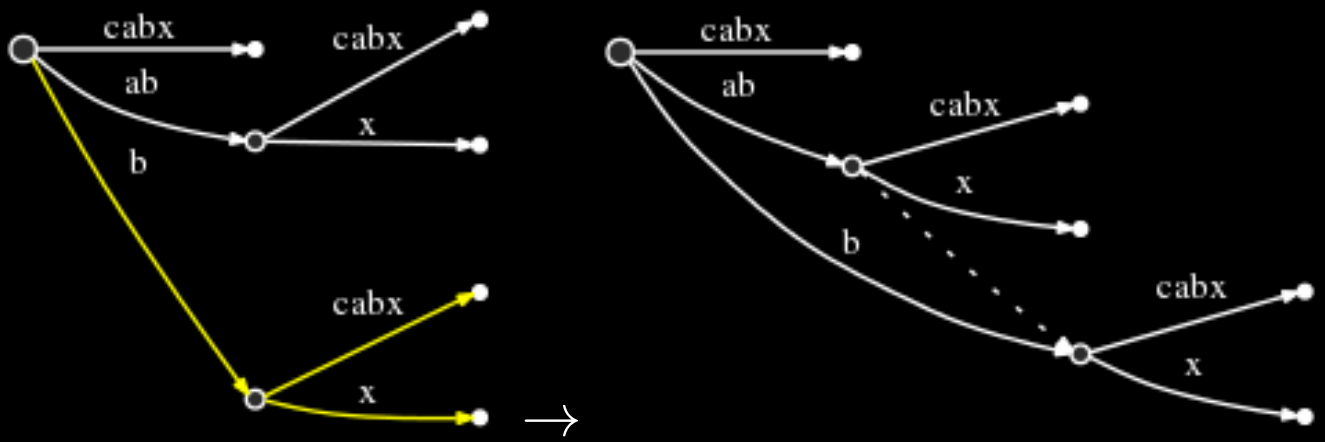
Należy zrobić jeszcze jedną rzecz. Mówi o tym ta reguła:

Reguła 2: Kiedy podzieliłem jakąś krawędź i dodałem do drzewa sufiksów nowy węzeł, i nie jest to pierwszy węzeł, który dodałem do drzewa w bieżącym kroku, łączę ten węzeł, który dodałem jako przedostatni, z tym węzłem, który dodałem jako ostatni, specjalnym łączem do sufiksu. Później będzie mowa o tym, do czego się przydają te łącza

Algorytm Ukkonena

abcabxabcd

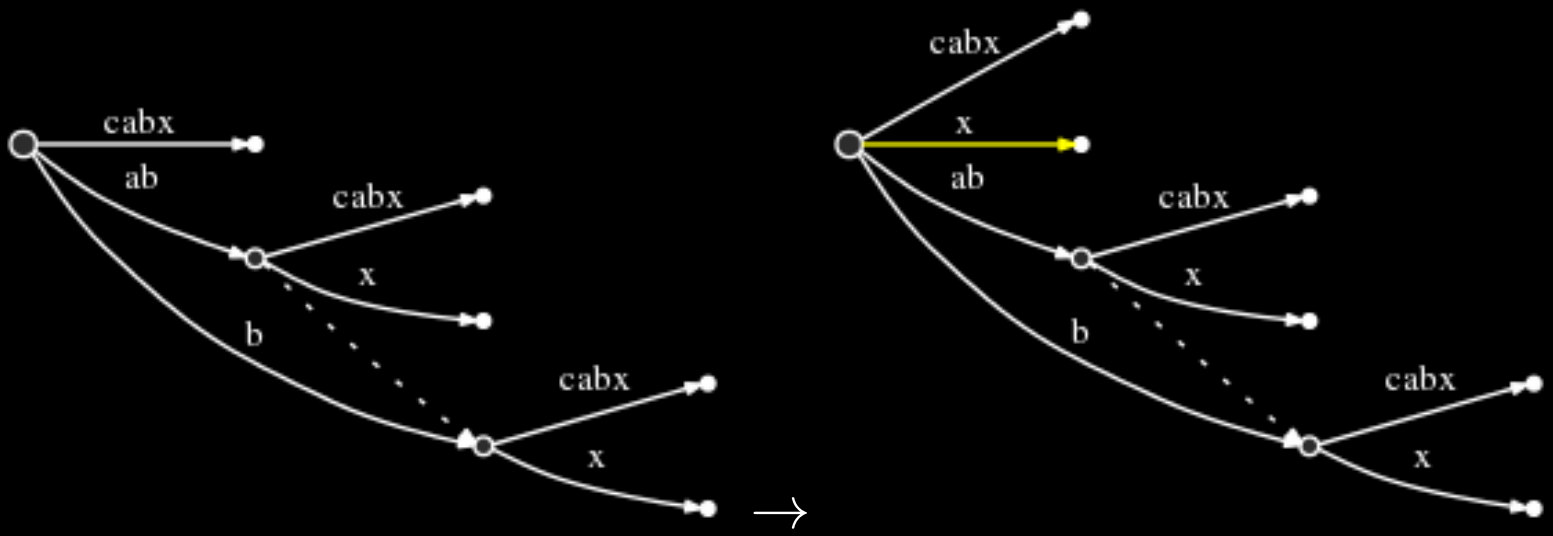
Kropkowana linia oznacza łącze do sufiksu:



Algorytm Ukkonena

`abcabxabcd`

Nadal chcę wstawić do drzewa sufiksów ostatni sufiks bieżącego kroku, czyli sufiks `x`. Wartość zmiennej `active_length` spadła do 0, więc wstawiam ten sufiks do korzenia drzewa. Z korzenia drzewa nie wychodzi żadna krawędź, która zaczynałaby się od znaku `x`, więc wstawiam do korzenia drzewa nową krawędź:



Algorytm Ukkonena

Krok 7: Zmieniam wartość zmiennej # na 7. W ten sposób dodaję znak a na końcu wszystkich tych krawędzi, które prowadzą do liści drzewa sufiksów. Potem próbuję wstawić nowy ostatni znak, czyli a, w `active_node`, czyli w węźle `root`. Z węzła `root` już wychodzi krawędź, która zaczyna się od znaku a, więc:

- Zmieniam aktywne położenie na (`root`, `'a'`, `1`)
- Kończę ten krok

Algorytm Ukkonena

Krok 8: Zmieniam wartość zmiennej # na 8. W ten sposób dodaję znak `b` na końcu wszystkich tych krawędzi, które prowadzą do liści drzewa sufiksów. Na krawędzi `ab`, która wychodzi z węzła `root`, za jej pierwszym znakiem już jest znak `b`, więc podobnie jak w poprzednich krokach zmieniam aktywne położenie na (`root`, `'a'`, `2`) i wartość zmiennej `backlog` na 2. W czasie $O(1)$ zauważam, że aktywne położenie znajduje się teraz na końcu krawędzi `ab`. Zatem zmieniam je na (`node_ab`, `'\0'`, `0`). Węzłem `node_ab` nazywam ten węzeł, do którego prowadzi z węzła `root` krawędź z etykietą `ab`

Algorytm Ukkonena

abcabxabcd

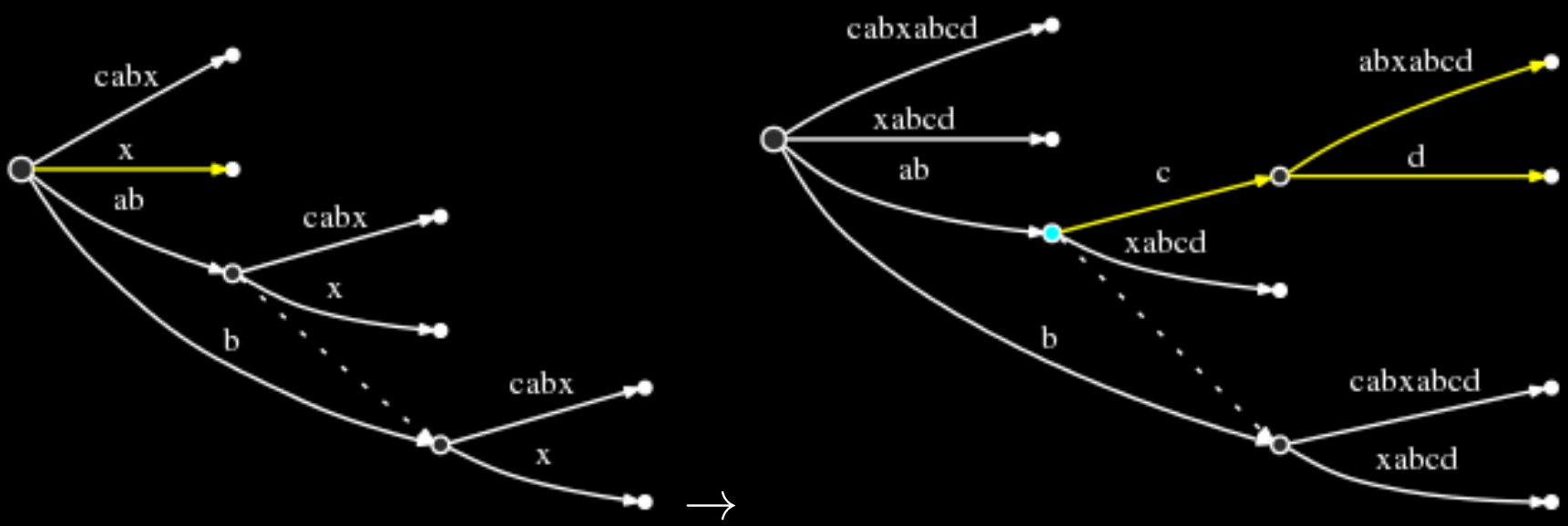
Krok 9: Należy wstawić do drzewa sufiksów znak c. Zmieniam wartość zmiennej # na 9. W ten sposób w czasie $O(1)$ dodaję znak c na końcu wszystkich tych krawędzi, które prowadzą do liści drzewa sufiksów. Z węzła node_ab już wychodzi krawędź, której etykieta zaczyna się od znaku c.
Zatem:

- Zmieniam aktywne położenie na (node_ab, 'c', 1)
- Zwiększam wartość zmiennej backlog o 1, do 4
- Kończę ten krok

Algorytm Ukkonena

abcabxabcd

Krok 10: Zwiększam wartość zmiennej # do 10. Zmienna backlog ma wartość 4, więc wstawiam do drzewa sufiksów sufiks abcd w aktywnym położeniu. Powoduje to podział krawędzi cabxabcd w czasie $O(1)$:



Algorytm Ukkonena

Węzeł `active_node` jest oznaczony kolorem. Oto ostatnia reguła:

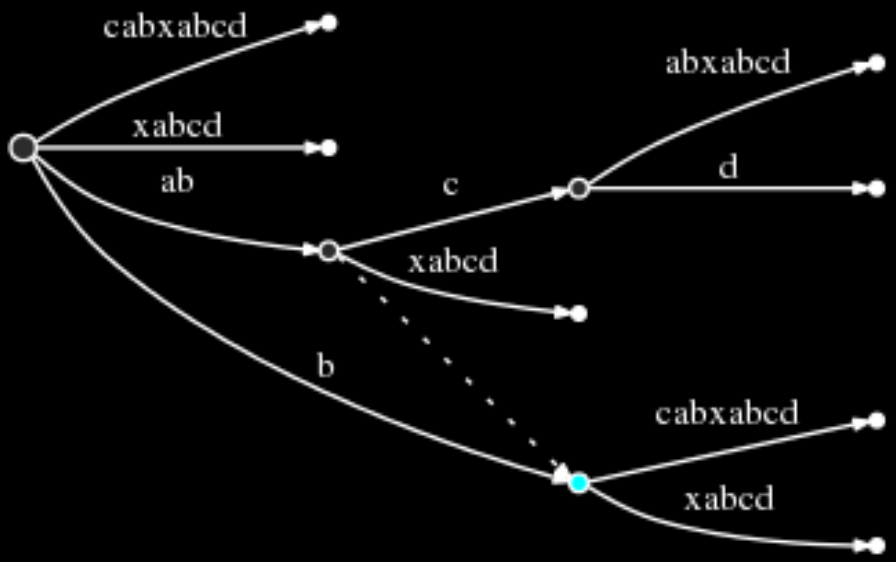
Reguła 3: Gdy podzieliłem krawędź od węzła `active_node`, który nie jest węzłem `root`:

- Jeśli z węzła `active_node` wychodzi łącze do sufiksu, to ustawiam węzeł `active_node` na ten węzeł, na który wskazuje to łącze do sufiksu
- Jeśli z węzła `active_node` nie wychodzi łącze do sufiksu, to ustawiam węzeł `active_node` na węzeł `root`
- Nie zmieniam wartości zmiennych `active_edge` i `active_length`

Algorytm Ukkonena

abcabxabcd

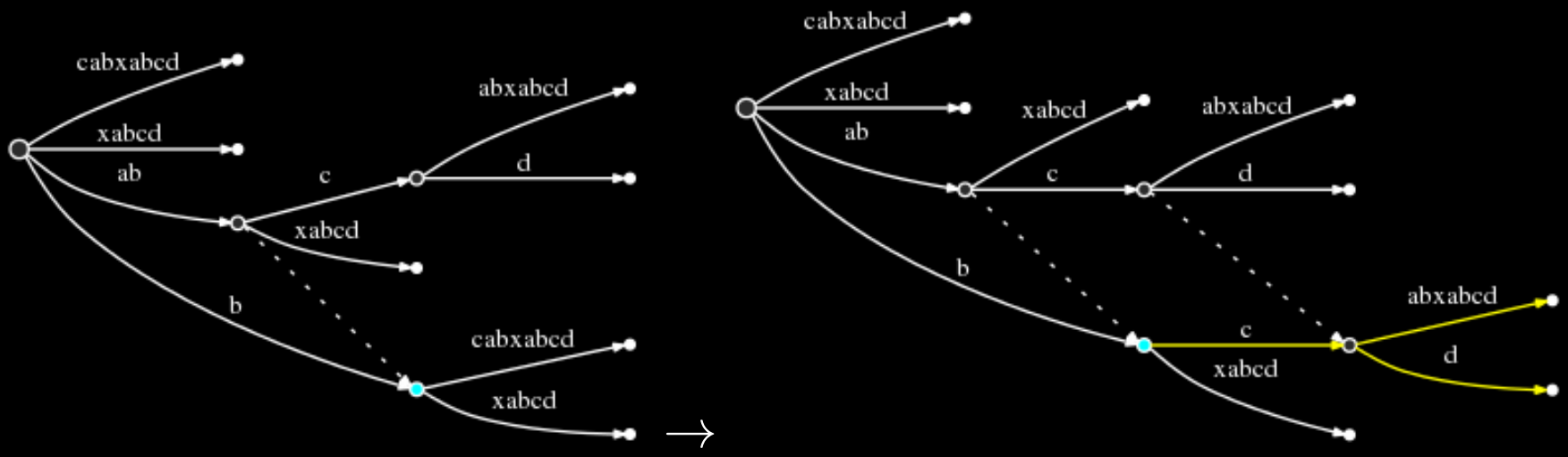
Aktywnym położeniem jest teraz (node_b, 'c', 1). Węzeł node_b jest oznaczony kolorem



Algorytm Ukkonena

abcabxabcd

Zmniejszam wartość zmiennej `backlog` do 3. Chcę dodać do drzewa sufiksów sufiks `bcd`. Dzięki regule 3 ustawiłem aktywne położenie na właściwy węzeł i krawędź, więc wystarczy wstawić ostatni znak tego sufiksu, czyli znak `d`, w bieżącym położeniu. Powoduje to kolejny podział krawędzi. Zgodnie z regułą 2 tworzę łącze do sufiksu od przedostatniego do ostatniego wstawionego węzła:



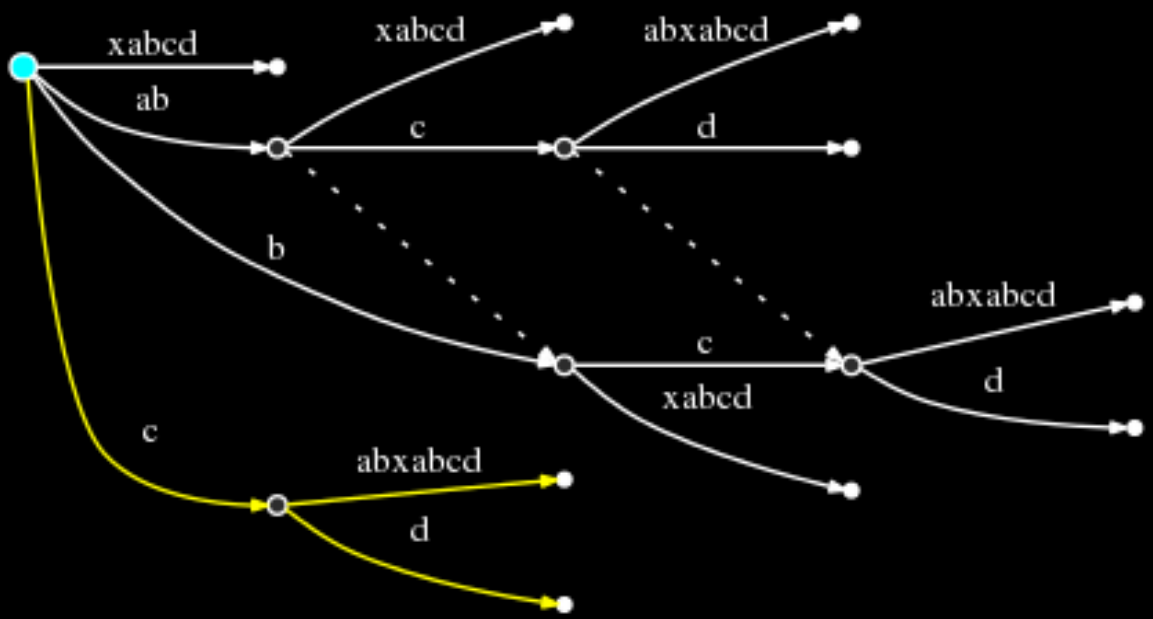
Algorytm Ukkonena

Dzięki łączom do sufiksów mogę zmieniać aktywne położenie w czasie $O(1)$

Algorytm Ukkonena

abcabxabcd

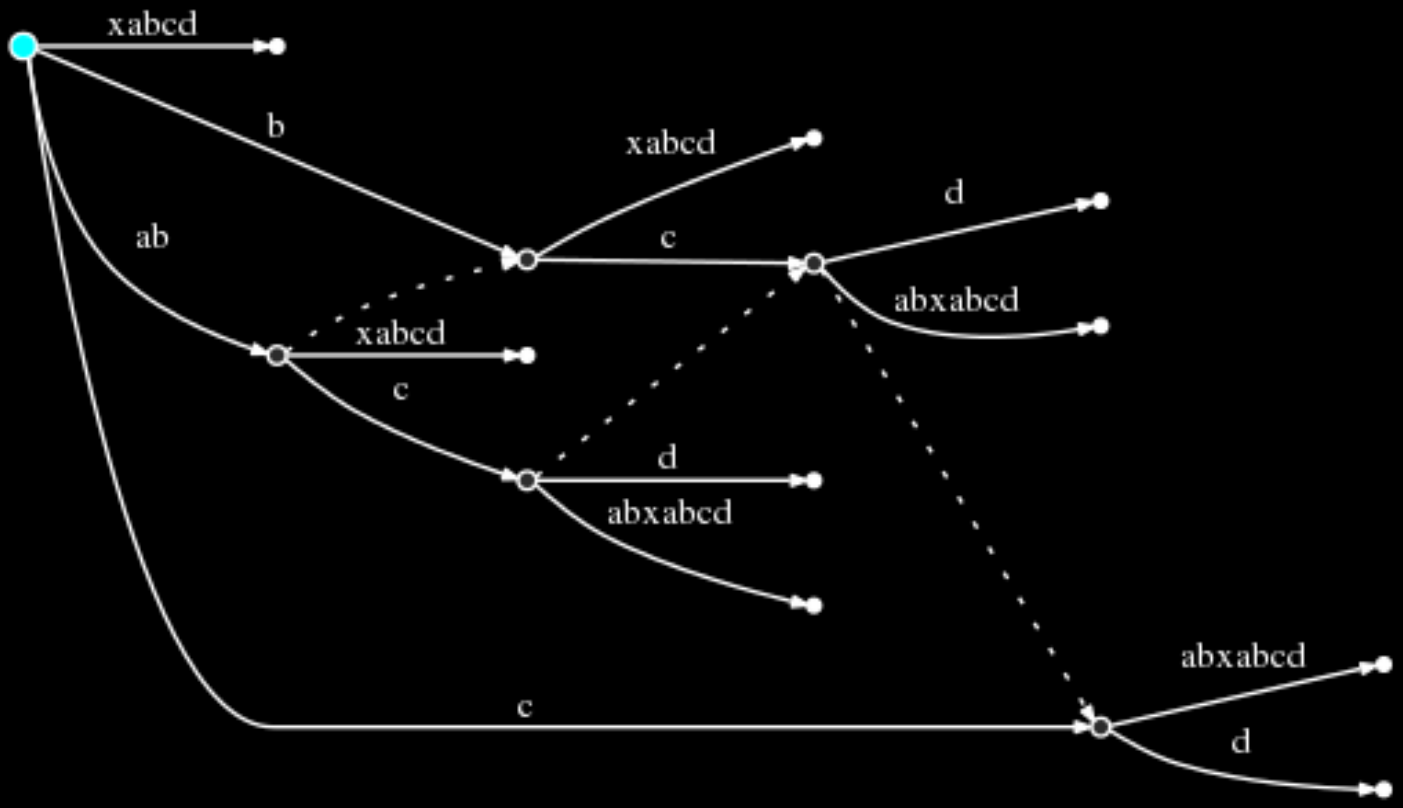
Krok 10 jeszcze się nie skończył. Zmienna `backlog` ma wartość 2. Trzeba skorzystać z reguły 3 i znowu zmienić aktywne położenie. Z węzła `active_node` nie wychodzi łącze do sufiksu, więc ustawiamy ten węzeł na węzeł `root`. Aktywne położenie to teraz (`root`, `c`, `1`). Wstawiam znak `d` za pierwszym znakiem tej krawędzi wychodzącej z węzła `root`, która ma etykietę `cabxabcd`. Powoduje to kolejny podział krawędzi:



Algorytm Ukkonena

abcabxabcd

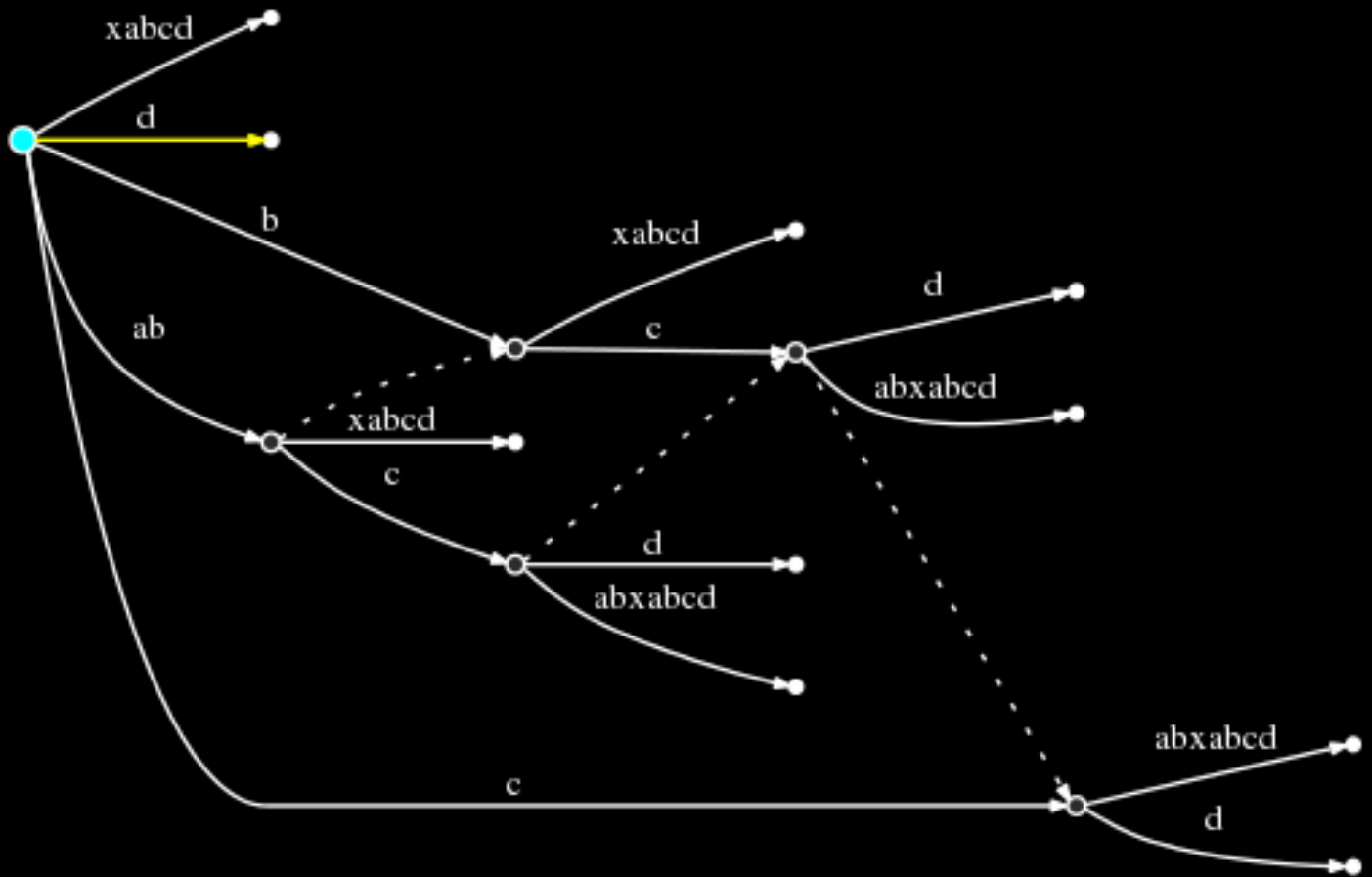
Stosuję regułę 2. Dodaję łącze do sufiksu do przedostatniego węzła utworzonego w tym kroku:



Algorytm Ukkonena

abcabxabcd

Teraz zmienna `backlog` ma wartość 1. Węzeł `active_node` to węzeł `root`. Korzystając z reguły 1 zmieniam aktywne położenie na (`root`, `'d'`, `0`). Ostatnie wstawienie w tym kroku to wstawienie sufiksu `d` w węźle `root`:



Algorytm Ukkonena

Gdy tekst ma n znaków, algorytm Ukkonena wykonuje n kroków. W każdym kroku albo tylko zmienia wartości zmiennych w czasie $O(1)$, albo ponadto dodaje do drzewa zaległe sufiksy, też w czasie $O(1)$. Zmienna `backlog` wskazuje, ile razy algorytm nie dodawał sufiksów do drzewa w poprzednich krokach. Ta zmienna zmniejsza się o 1 zawsze wtedy, kiedy algorytm dodaje sufiks do drzewa. Zatem algorytm Ukkonena dodaje do drzewa sufiksy $O(n)$ razy. Zatem całkowita złożoność czasowa algorytmu Ukkonena to $O(n)$.

Podsumowanie

Podsumowanie

- Tablice sufiksów
- Zastosowania tablic sufiksów
- Drzewa sufiksów
- Zastosowania drzew sufiksów
- Algorytm Ukkonena budowania drzew sufiksów

Pomysły, uwagi, pytania, sugestie

Proszę wysyłać podpisane pomysły, uwagi, pytania, sugestie na temat wykładów lub laboratoriów na adres mgc@agh.edu.pl

lub wpisywać anonimowe pomysły, uwagi, pytania, sugestie pod adresem https://tiny.cc/algorytmy-tekstowe

Do zobaczenia
na następnym wykładzie

Jego tematem będą pomysłowe algorytmy tekstowe

Źródła ilustracji

Zdeněk Jirotka, Saturnin – zrzut ekranu

https://commons.wikimedia.org/wiki/File:DNA123_rotated.png

https://classic.csunplugged.org/activities/text-compression/

Marshall Custiss Hazard, A Complete Concordance to the American Standard Version of the Holy Bible – zrzut ekranu z Archive.org

Donald E. Knuth, The Art of Computer Programming, tom 3

https://en.wikipedia.org/wiki/Udi_Manber

https://en.wikipedia.org/wiki/Eugene_Myers

https://www.linkedin.com/in/pgweiner/

https://www.cs.helsinki.fi/u/ukkonen/

https://stackoverflow.com/questions/9452701/ukkonens-suffix-tree-algorithm-in-plain-english

Źródła zdjęć