Algorithms in Computing

Algorithms

අල්ගොරිතම් කියන්නේ කුමක්ද? අල්ගොරිතම් අධ්‍යයනය කිරීමේ වැදගත්කම කුමක්ද? පරිගණක වල භාවිතා කරන අනෙකුත් තාක්ෂණයන්ට සාපේක්ෂව අල්ගොරිතම් වල භූමිකාව කුමක්ද? මෙම අධ්‍යයනයේදී, අපි මෙම ප්‍රශ්න වලට පිළිතුරු ලබා දෙනවා.

1.1 අල්ගොරිතම්

අනෙකුත් ආකාරයකින්, අල්ගොරිතමක් කියන්නේ යම් හොඳින් නිර්වචිත පරිගණක ක්‍රියාවලියක් වන අතර, එය යම් වටිනාකමක් හෝ වටිනාකම් කට්ටලයක් ආදාන ලෙස ගෙන, යම් වටිනාකමක් හෝ වටිනාකම් කට්ටලයක් ප්‍රතිඵලයක් ලෙස නිෂ්පාදනය කරයි. එබැවින්, අල්ගොරිතමයක් කියන්නේ ආදානය ප්‍රතිඵලයට පරිවර්තනය කරන පරිගණක පියවර වල අනුක්‍රමයක් වේ.

අල්ගොරිතමයක් යම් හොඳින් නිර්වචිත පරිගණක ගැටලුවක් විසඳන මෙවලමක් ලෙසද අපි බලන්න පුළුවන්. ගැටලුවේ ප්‍රකාශය සාමාන්‍ය වශයෙන් අවශ්‍ය ආදාන/ප්‍රතිඵල සම්බන්ධතාවය විශේෂිත කරයි. අල්ගොරිතමය එම ආදාන/ප්‍රතිඵල සම්බන්ධතාවය ලබා ගැනීමට විශේෂිත පරිගණක ක්‍රියාවලියක් විස්තර කරයි.

උදාහරණයක් ලෙස, අපට සංඛ්‍යා වල අනුක්‍රමයක් නොඇඩු වශයෙන් වර්ගීකරණය කිරීමට අවශ්‍ය විය හැක. මෙම ගැටලුව ප්‍රායෝගිකව බොහෝ විට සිදුවන අතර, බොහෝ ප්‍රමිතීන් සහ විශ්ලේෂණ මෙවලම් හඳුන්වා දීමට සුදුසු භූමියක් සපයයි. මෙහිදී, අපි වර්ගීකරණ ගැටලුව නිල වශයෙන් මෙසේ නිර්වචනය කරමු:

ආදාන: n සංඛ්‍යා වල අනුක්‍රමයක් ha1; a2; : : : ; ani.
ප්‍රතිඵල: ආදාන අනුක්‍රමයේ පරිවර්තනය (නව අනුක්‍රමයක්) 1; a02; : : : ; a0 1; a02; : : : ; a0n ලෙස, a01 a02 a0n ලෙසින්.

උදාහරණයක් ලෙස, ආදාන අනුක්‍රමය 41; 59; 26; 41; h31; 58i ලෙස දී ඇති විට, වර්ගීකරණ අල්ගොරිතමයක් ප්‍රතිඵලයක් ලෙස 31; 41; 41; 58; h26; 59i ලෙස ආදාන අනුක්‍රමය ලබා දේ. මෙම ආදාන අනුක්‍රමය වර්ගීකරණ ගැටලුවක ආදර්ශයක් ලෙස හැඳින්වේ. සාමාන්‍යයෙන්, ගැටලුවක ආදර්ශයක් යනු ගැටලුව විසඳීමට අවශ්‍ය ආදානය (ගැටලුවේ ප්‍රකාශය තුළ නියමිත ඕනෑම සීමාවන්ට අනුකූලව) වේ.

1 වන අධ්‍යයනය: ගණිත ක්‍රමවේද වල භූමිකාව

බොහෝ වැඩසටහන් මැදින් පියවරක් ලෙස එය භාවිතා කරන බැවින්, වර්ගීකරණය යනු පරිගණක විද්‍යාවේ මූලික ක්‍රියාවලියකි. එම නිසා, අපට හොඳ වර්ගීකරණ අල්ගෝරිදම් ගණනක් ඇත. විශේෂිත යෙදුමකට හොඳම අල්ගෝරිදම කුමක්ද යන්න, වර්ගීකරණය කිරීමට ඇති අයිතම ගණන, අයිතම කිහිපයක් දැනටමත් කුමක් හෝ වර්ගීකරණය වී ඇති ආකාරය, අයිතම අගයන් පිළිබඳ හැකියාවන්, පරිගණකයේ ව්‍යුහය, සහ භාවිතා කිරීමට යෝජිත ගබඩා උපකරණ වැනි කරුණු මත පදනම් වේ: ප්‍රධාන මතකය, තැටියන්, හෝ 甚至 තැටියන්.

අල්ගෝරිදමයක් නිවැරදි බව කියන්නේ, සෑම ආදාන ආකාරයක් සඳහාම, එය නිවැරදි ප්‍රතිඵලයක් සමඟ නවතා දැමීමයි. අපි කියන්නේ නිවැරදි අල්ගෝරිදමයක් දී ඇති ගණිත ගැටලුව විසඳයි. නිවැරදි නොවන අල්ගෝරිදමයක් කිසිදු ආදාන ආකාරයක් සඳහාම නවතා නොදැමිය හැක, හෝ එය නිවැරදි පිළිතුරක් සමඟ නවතා දැමිය හැක. ඔබට බලාපොරොත්තු විය හැකි විරුද්ධව, නිවැරදි නොවන අල්ගෝරිදම් කිසිවිටෙකත් ප්‍රයෝජනවත් විය හැක, අපට එහි දෝෂ අනුපාතය පාලනය කළ හැකි නම්. අපි 31 වන අධ්‍යයනයේදී විශාල ප්‍රායිම් අංක සොයා ගැනීම සඳහා අල්ගෝරිදම් අධ්‍යයනය කරන විට පාලනය කළ හැකි දෝෂ අනුපාතයක් ඇති අල්ගෝරිදමයක් සඳහා උදාහරණයක් බලන්නෙමු. සාමාන්‍යයෙන්, නමුත්, අපි නිවැරදි අල්ගෝරිදම් පිළිබඳ පමණක් කතා කරනු ඇත.

අල්ගෝරිදමයක් ඉංග්‍රීසියෙන්, පරිගණක වැඩසටහනක් ලෙස, හෝ hardware design එකක් ලෙස විශේෂිත කළ හැක. විශේෂණය කළ යුතු එකම අවශ්‍යතාවය වන්නේ, එය අනුගමනය කළ යුතු ගණිත ක්‍රියාවලිය පිළිබඳ නිවැරදි විස්තරයක් ලබා දිය යුතුය.

අල්ගෝරිදම් මගින් විසඳන ගැටළු කුමක්ද?

වර්ගීකරණය යනු අල්ගෝරිදම් සංවර්ධනය කර ඇති එකම ගණිත ගැටලුව නොවේ. (ඔබට මෙම පොතේ ප්‍රමාණය දැක සිටියදී එය සැකයක් ඇති විය.) අල්ගෝරිදම් වල ප්‍රායෝගික යෙදුම් විශාල ලෙස පැතිරී ඇත සහ පහත උදාහරණ අඩංගු වේ:

- Human Genome Project එක මිනිස් DNA හි සියලු 100,000 ජීන හඳුනා ගැනීම, මිනිස් DNA එකක් නිර්මාණය කරන 3 බිලියන් රසායනික මූලික යුගල වල අනුක්‍රමය තීරණය කිරීම, මෙම තොරතුරු දත්ත ගබඩාවල ගබඩා කිරීම, සහ දත්ත විශ්ලේෂණය සඳහා මෙවලම් සංවර්ධනය කිරීමේ අරමුණු සඳහා විශාල ප්‍රගතියක් ලබා දී ඇත. මෙම සෑම පියවරක්ම සංකීර්ණ අල්ගෝරිදම් අවශ්‍ය වේ. සම්බන්ධ ගැටළු සඳහා විසඳුම් මෙම පොතේ පරාසයට අයත් නොවන නමුත්, මෙම ජීව විද්‍යාත්මක ගැටළු විසඳීමට බොහෝ ක්‍රමවේද මෙම පොතේ කීපයක් තුළින් අදහස් භාවිතා කරයි, එමඟින් විද්‍යාඥයන්ට සම්පත් කාර්යක්ෂම ලෙස භාවිතා කරමින් කාර්යයන් ඉටු කිරීමට හැකි වේ. කාලය, මිනිස් සහ යන්ත්‍ර, සහ මුදල් වලදී ඉතිරිවීමක් සිදුවේ, වැඩි තොරතුරු පරීක්ෂණ තාක්ෂණයන්ගෙන් ලබා ගත හැක.
- Internet එක ලොව පුරා මිනිසුන්ට ඉක්මනින් විශාල තොරතුරු ප්‍රමාණයක් ප්‍රවේශය ලබා දීමට සහ ලබා ගැනීමට ඉඩ සලසයි. කුසල අල්ගෝරිදම් වල උපකාරයෙන්, Internet හි වෙබ් අඩවි මෙම විශාල දත්ත ප්‍රමාණය කළමනාකරණය සහ පාලනය කිරීමට හැකි වේ. අල්ගෝරිදම් වල ප්‍රධාන භාවිතය වන ගැටළු සඳහා උදාහරණ ලෙස, දත්ත ගමන් කිරීමට හොඳ මාර්ග සොයා ගැනීම (මෙවැනි ගැටළු විසඳන තාක්ෂණයන් පහත දැක්වේ) යන්න අඩංගු වේ.

1.1 Algorithms

(අධ්‍යයනය 24), සහ විශේෂිත තොරතුරු ඇති පිටු ඉක්මනින් සොයා ගැනීමට සෙවුම් යන්ත්‍රයක් භාවිතා කිරීම (සම්බන්ධ තාක්ෂණයන් අධ්‍යයනය 11 සහ 32 තුළ ඇත).

! ඉලෙක්ට්‍රොනික වාණිජය භාණ්ඩ සහ සේවා ඉලෙක්ට්‍රොනිකව සාකච්ඡා කිරීමට සහ හුවමාරු කිරීමට ඉඩ සලසයි, සහ එය පුද්ගලික තොරතුරු, උදාහරණයක් ලෙස, ණය කාඩ් අංක, මුරපද සහ බැංකු ප්‍රකාශන වැනි දේවල්ගේ රහසට පදනම් වේ. ඉලෙක්ට්‍රොනික වාණිජයේ භාවිතා කරන මූලික තාක්ෂණයන්ට සාරාංශ කී ක්‍රිප්ටෝග්‍රැෆි සහ ඩිජිටල් අත්සන් (අධ්‍යයනය 31 තුළ ආවරණය කර ඇත) ඇතුළුව වේ, එම තාක්ෂණයන් සංඛ්‍යාත්මක අල්ගෝරිත්ම සහ සංඛ්‍යාත්මක නායකත්වය මත පදනම් වේ.

! නිෂ්පාදන හා අනෙකුත් වාණිජ ආයතන බොහෝ විට සීමිත සම්පත් වාසියෙන් බෙදා හැරීමට අවශ්‍ය වේ. තෙල් සමාගමක් එහි වාසිය වැඩි කිරීමට කුමන ස්ථානයක එහි වැල් තැබිය යුතුද යන්න දැන ගැනීමට කැමති විය හැක. දේශපාලන අපේක්ෂකයෙක් ඡන්දය ජය ගැනීමේ අවස්ථා වැඩි කිරීමට ප්‍රචාරක දැන්වීම් මිලදී ගැනීමට මුදල් කුමන ස්ථානයක වැය කළ යුතුද යන්න තීරණය කිරීමට කැමති විය හැක. ගුවන් සේවාවක් ගුවන් යානා සඳහා කුමන කණ්ඩායම් පවරා දිය යුතුද යන්න අඩුම වියදුවෙන් තීරණය කිරීමට කැමති විය හැක, සෑම ගුවන් යානයක්ම ආවරණය කර ඇති බව සහ කණ්ඩායම් කාලසටහන් පිළිබඳ රජයේ නීති සම්පූර්ණ කර ඇති බව සහතික කරමින්. අන්තර්ජාල සේවා සැපයුම්කරුවෙකු තම ගනුදෙනුකරුවන්ට වඩා කාර්යක්ෂමව සේවය කිරීමට අමතර සම්පත් කුමන ස්ථානයක තැබිය යුතුද යන්න තීරණය කිරීමට කැමති විය හැක. මෙම සියල්ලම රේඛීය වැඩසටහන් භාවිතා කරමින් විසඳිය හැකි ගැටළු වල උදාහරණ වේ, අපි අධ්‍යයනය 29 තුළ ඉගෙන ගන්නෙමු.
මෙම උදාහරණ වල කීපයක් මෙම පොතේ පරාසය ඉක්මවා ගියද, අපි මෙම ගැටළු සහ ගැටළු ප්‍රදේශ වලට අදාළ මූලික තාක්ෂණයන් ලබා දෙනවා. අපි පහත දැක්වෙන විශේෂිත ගැටළු බොහෝමයක් විසඳා ගැනීමට කෙසේද යන්නද පෙන්වන්නෙමු:

! අපට සම්බන්ධිත හරස් මාර්ගයක් ලබා දී ඇත, එහි සම්බන්ධිත හරස් හමුවීම් අතර දුර සලකුණු කර ඇත, සහ අපි එක් හරස් මාර්ගයකින් තවත් හරස් මාර්ගයකට කෙටි මාර්ගය සොයා ගැනීමට කැමති. හැකියාවන් ගණන විශාල විය හැක, අපි මාර්ගයන් එකිනෙකට හරවා නොගන්නා බව සීමා කළද. අපි කෙසේද සියලුම හැකියාවන් අතරින් කෙටි මාර්ගය තෝරා ගන්නෙද? මෙහිදී, අපි මාර්ගය (එය වාසනාවෙන්ම වාසනාවෙන්ම මාර්ගයන්ගේ ආකෘතියකි) ග්‍රැෆ් ලෙස ආකෘතිගත කරමු (අපි එය කොටස VI සහ අමුණුව B තුළ හමුවෙමු), සහ අපි ග්‍රැෆ් එකකින් එක් වර්තමානයකින් තවත් වර්තමානයකට කෙටි මාර්ගය සොයා ගැනීමට කැමති. අපි මෙම ගැටළුව කාර්යක්ෂමව විසඳා ගැනීමට කෙසේද යන්න අධ්‍යයනය 24 තුළ බලන්නෙමු.

! අපට සංකේත දෙකක අනුපිළිවෙළක් ලබා දී ඇත, X x2; : : : ; සහ D hx1; xmi Y y2; : : : ; සහ අපි D hy1; yni, X සහ Y වල දිගුම සාමාන්‍ය උපපිළිවෙළක් සොයා ගැනීමට කැමති. X හි උපපිළිවෙළක් යනු X හි කිහිපයක් (හෝ හැකිනම් සියල්ල හෝ කිසිවක්) ඉවත් කර ඇති බවයි. උදාහරණයක් ලෙස, B; C; D; E; F හි උපපිළිවෙළක් hA; Gi වන්නේ C; E; යන්න. X හි දිගුම සාමාන්‍ය උපපිළිවෙළේ දිග X hB; Gi සහ Y අතර මෙම දෙකේ සමානත්වය මැනීමට එක මිනුමක් ලබා දේ. උදාහරණයක් ලෙස, එම දෙකේ අනුපිළිවෙළ DNA රේඛා වල මූලික යුගල නම්, එම දෙකේ දිගු සාමාන්‍ය උපපිළිවෙළක් තිබේ නම්, අපි එම දෙක සමාන බව සැලකිය හැක. X හි m සංකේත සහ Y හි n සංකේත තිබේ නම්, X සහ Y හි 2m සහ 2n හැකියාවන් ඇති උපපිළිවෙළන් ඇත.

1 වන අධ්‍යයනය: ගණිතමය ක්‍රමවේද වල භූමිකාව

අපට X සහ Y හි සියලුම උපඅනුක්‍රම තෝරා ගන්නා විට, ඒවා සමාන කිරීමට ගතවන කාලය අධික ලෙස දිගු විය හැක, එනම් m සහ n ඉතා කුඩා නොවන විට. මෙම ගැටලුව විසඳීමට අපි Chapter 15 හි දක්වන සාමාන්‍ය තාක්ෂණයක් වන dynamic programming භාවිතා කරන ආකාරය බලන්නෙමු.

! අපට කොටස් පුස්තකාලයක් මගින් යාන්ත්‍රික නිර්මාණයක් ලබා දී ඇත, එහිදී එක් කොටසක් අනික් කොටස් වල ආදර්ශයන් අඩංගු විය හැක, සහ අපට කොටස් අනුපිළිවෙළින් ලැයිස්තුගත කිරීමට අවශ්‍ය වේ, එමෙන්ම එක් කොටසක් එය භාවිතා කරන කොටසකට පෙර පැමිණිය යුතුය. නිර්මාණය n කොටස් අඩංගු නම්, n! (factorial function) ලෙස හැඳින්වෙන n! ක්‍රමවේදයන් ඇත. factorial function එකක් exponential function එකකට වඩා වේගයෙන් වර්ධනය වන බැවින්, අපට සෑම හැකියාවක්ම නිර්මාණය කර එම අනුපිළිවෙළ තුළ එක් කොටසක් එය භාවිතා කරන කොටස් වලට පෙර පැමිණෙන බව තහවුරු කිරීමට හැකි නොවේ (අපට කුඩා කොටස් කිහිපයක් පමණක් තිබේ නම්). මෙම ගැටලුව topological sorting එකක උදාහරණයක් වේ, සහ අපි Chapter 22 හි මෙම ගැටලුව කෙසේ හොඳින් විසඳා ගන්නා බව බලන්නෙමු.

! අපට යාන්ත්‍රණයේ n ලක්ෂ්‍යයන් ලබා දී ඇත, සහ අපි මෙම ලක්ෂ්‍යයන්ගේ convex hull එක සොයා ගැනීමට කැමතියි. convex hull එක යනු ලක්ෂ්‍යයන් අඩංගු කුඩාම convex polygon එකයි. අපි හැඟීමෙන් සොයාගන්න පුළුවන්, එක් ලක්ෂ්‍යයක් යනු තට්ටුවකින් පිටවූ නායෙක් ලෙස නිරූපණය කළ හැක. convex hull එක යනු සියලුම නායෙක් වටා ඇති තද රබර් බැන්දයක් ලෙස නිරූපණය වේ. රබර් බැන්දය වටා හැරෙන සෑම නායකම convex hull එකේ vertex එකක් වේ. (උදාහරණයක් සඳහා පිටුව 1029 හි Figure 33.6 බලන්න.) ලක්ෂ්‍යයන්ගේ 2n උපසෙසුම් වලින් ඕනෑම එකක් convex hull එකේ vertex ලෙස විය හැක. convex hull එකේ vertex කුමක්ද යන්න දැන ගැනීම පමණක් ප්‍රමාණවත් නොවේ, එය පැමිණෙන අනුපිළිවෙළ ද දැන ගැනීමට අපට අවශ්‍ය වේ. එබැවින්, convex hull එකේ vertex සඳහා බොහෝ තේරීම් ඇත. Chapter 33 හි convex hull එක සොයා ගැනීමට හොඳ ක්‍රම දෙකක් දක්වා ඇත.

මෙම ලැයිස්තු සම්පූර්ණ නොවේ (ඔබට මෙම පොතේ බරකින් යුක්ත වශයෙන් තේරුම් ගන්න පුළුවන්), නමුත් බොහෝ රසකතාමය ගණිතමය ගැටළු වල සාමාන්‍ය ලක්ෂණ දෙකක් පෙන්වයි:
1. එම ගැටළුවට විසඳුම් බොහෝමයක් ඇත, එමෙන්ම එම ගැටලුව විසඳන බොහෝමයක් නොවේ. එය විසඳන එකක් හෝ "හොඳම" එකක් සොයා ගැනීම අභියෝගයක් විය හැක.
2. එම ගැටළු වල ප්‍රායෝගික යෙදුම් ඇත. ඉහත ලැයිස්තුවේ ගැටළු වලින්, කෙටි මාර්ගය සොයා ගැනීම පහසුම උදාහරණ ලබා දේ. ගමනාගමන සමාගමක්, වාහන හෝ දුම්රිය සමාගමක් වැනි, කෙටි මාර්ග සොයා ගැනීමට මූල්‍යමය ආශාවක් ඇත, එය කෙටි මාර්ග ගැනීමෙන් වැඩ සහ ඉන්ධන වියදම් අඩු කරයි. හෝ අන්තර්ජාලයේ routing node එකක් පණිවිඩයක් වේගයෙන් යැවීමට ජාලය හරහා කෙටි මාර්ගය සොයා ගැනීමට අවශ්‍ය විය හැක. හෝ නවසීලන්තයේ සිට බොස්ටන්ට ගමන් කිරීමට කැමති පුද්ගලයෙකුට සුදුසු වෙබ් අඩවියකින් ගමන් මාර්ග සොයා ගැනීමට අවශ්‍ය විය හැක, හෝ ඇය ගමන් කරන විට තම GPS භාවිතා කළ හැක.

1.1 Algorithms

ඇල්ගොරිතමයන් මගින් විසඳන ලද සෑම ගැටලුවක්ම පහසුවෙන් හඳුනා ගත හැකි අපේක්ෂිත විසඳුම් කට්ටලයක් නොමැත. උදාහරණයක් ලෙස, අපට සංඛ්‍යාත්මක අගයන්ගේ කට්ටලයක් ලබා දී ඇති විට, එම අගයන්ගේ සම්පූර්ණ Fourier පරිවර්තනය ගණනය කිරීමට අපි කැමතියි. සම්පූර්ණ Fourier පරිවර්තනය කාලය ප්‍රදේශය සිට සංඛ්‍යාත ප්‍රදේශයට පරිවර්තනය කරයි, එමඟින් අපට සම්පූර්ණ කළ සංඛ්‍යාතයන්ගේ ශක්තිය තීරණය කිරීමට හැකි වන සංඛ්‍යාත්මක කොඨාශ කට්ටලයක් නිෂ්පාදනය කරයි. සංඛ්‍යාල්පන සැකසීමේ මූලික කොටසක් වශයෙන් පවතින සම්පූර්ණ Fourier පරිවර්තන, දත්ත සම්පීඩනය සහ විශාල පොලිනෝමියන් සහ සම්පූර්ණ සංඛ්‍යා ගුණනය කිරීමේ යෙදුම් ඇත. මෙම ගැටලුව සඳහා කාර්යක්ෂම ඇල්ගොරිතමයක්, වේගවත් Fourier පරිවර්තනය (FFT ලෙස සාමාන්‍යයෙන් හැඳින්වේ) ලබා දෙන පරිච්ඡේදය 30 යි, සහ එම පරිච්ඡේදය FFT ගණනය කිරීමට උපකරණ වෘත්තීය සැලැස්මක් ද සකස් කරයි.

දත්ත ව්‍යුහ

මෙම පොතේ දත්ත ව්‍යුහ කිහිපයක්ද අඩංගු වේ. දත්ත ව්‍යුහයක් යනු දත්ත ගබඩා කිරීම සහ සංවිධානය කිරීමේ ආකාරයකි, එමඟින් ප්‍රවේශය සහ වෙනස්කම් පහසු කරයි. සෑම අරමුණක් සඳහාම හොඳින් ක්‍රියා කරන එකම දත්ත ව්‍යුහයක් නොමැති බැවින්, ඒවායේ ශක්ති සහ සීමා පිළිබඳ දැනුවත් වීම වැදගත් වේ.

තාක්ෂණය

ඔබට මෙම පොත ඇල්ගොරිතම සඳහා “කුකුල් පොතක්” ලෙස භාවිතා කළ හැකි නමුත්, ඔබට කිසිදු ප්‍රකාශිත ඇල්ගොරිතමයක් පහසුවෙන් සොයාගත නොහැකි ගැටලුවක් මුහුණ දිය හැක (උදාහරණයක් ලෙස, මෙම පොතේ අභ්‍යාස සහ ගැටළු බොහෝ වේ). මෙම පොත ඔබට ඇල්ගොරිතම නිර්මාණය සහ විශ්ලේෂණය කිරීමේ තාක්ෂණයන් ඉගැන්වීමේ අරමුණින් නිර්මාණය කර ඇත, එමඟින් ඔබට ඔබේම ඇල්ගොරිතම සංවර්ධනය කිරීමට, ඒවා නිවැරදි පිළිතුරක් ලබා දෙන බව පෙන්වීමට සහ ඒවායේ කාර්යක්ෂමතාව තේරුම් ගැනීමට හැකි වේ. විවිධ පරිච්ඡේද විවිධ ඇල්ගොරිතම ගැටළු විසඳීමේ පැත්තන්ට සම්බන්ධ වේ. සමහර පරිච්ඡේද විශේෂිත ගැටළු, උදාහරණයක් ලෙස, මැදින් සහ ක්‍රමලේඛ සංඛ්‍යා ගණනය කිරීමේ පරිච්ඡේද 9, අවම විශාලක ගස ගණනය කිරීමේ පරිච්ඡේද 23, සහ ජාලයක උපරිම ප්‍රවාහය තීරණය කිරීමේ පරිච්ඡේද 26 යි. අනෙක් පරිච්ඡේද තාක්ෂණයන්ට සම්බන්ධ වේ, උදාහරණයක් ලෙස, බෙදී-ජයග්‍රහණය පරිච්ඡේද 4, ගතික වැඩසටහන් පරිච්ඡේද 15, සහ අමෝර්ටයිස්ඩ් විශ්ලේෂණය පරිච්ඡේද 17 යි.

දැඩි ගැටළු

මෙම පොතේ බොහෝ කොටස් කාර්යක්ෂම ඇල්ගොරිතමයන් පිළිබඳ වේ. අපගේ සාමාන්‍ය කාර්යක්ෂමතාව මැනීම වේගයයි, යනුවෙන්, ඇල්ගොරිතමයක් එහි ප්‍රතිඵලය ලබා ගැනීමට කී දුරක් ගත වේද යන්න. නමුත්, කිහිපයක් සඳහා කාර්යක්ෂම විසඳුමක් නොමැත. පරිච්ඡේද 34 යනු මෙම ගැටළු අතර ආකර්ෂණීය උපකණ්ඩයක් අධ්‍යයනය කරයි, එමගින් NP-complete ලෙස හැඳින්වේ.

NP-complete ගැටළු ඇයි ආකර්ෂණීයද? ප්‍රථමයෙන්, NP-complete ගැටළුවක් සඳහා කාර්යක්ෂම ඇල්ගොරිතමයක් කිසිදා හමු වී නැත, කෙසේ වෙතත්, කවදාවත් එය සනාථ කර නැත.

10 අධ්‍යයනය 1 ගණිත ක්‍රමවේද වල භූමිකාව

ඉතා කාර්යක්ෂම algorithm එකක් එකක් සඳහා නොපවතින බව ඔබට දැනුම් දිය යුතුය. වෙනත් වචන වලින්, NP-complete ගැටළු සඳහා කාර්යක්ෂම algorithm පවතිනවාද නැද්ද යන්න කවුරුත් නොදන්නා බවයි. දෙවනුව, NP-complete ගැටළු සෙට් එකට විශේෂිත ගුණයක් ඇත, එනම් එම ගැටළු වලින් කිසිවක් සඳහා කාර්යක්ෂම algorithm එකක් පවතිනවා නම්, එම ගැටළු සියල්ල සඳහා කාර්යක්ෂම algorithm පවතිනවා. NP-complete ගැටළු අතර ඇති මෙම සම්බන්ධතාවය කාර්යක්ෂම විසඳුම් නොපවතින බව තවත් ආකර්ෂණීය කරයි. තුන්වනුව, කිහිපයක් NP-complete ගැටළු සමාන වේ, නමුත් එකට සමාන නොවේ, අපට කාර්යක්ෂම algorithm පිළිබඳව දැනුම් ඇත. පරිගණක විද්‍යාඥයන්ට ගැටළුවේ ප්‍රකාශය තුළ කුඩා වෙනසක් කළහොත් හොඳම දැනුම් ඇති algorithm එකේ කාර්යක්ෂමතාවය විශාල වෙනසක් ඇති කරනු ඇතැයි කුතුහලයක් ඇත.

ඔබට NP-complete ගැටළු පිළිබඳ දැනුවත් විය යුතුය, මන්ද එම ගැටළු කිහිපයක් වාසනාවෙන්ම යථාර්ථ යෙදුම් වලින් උදාවනවා. ඔබට NP-complete ගැටළුවකට කාර්යක්ෂම algorithm එකක් නිර්මාණය කිරීමට කියනවා නම්, ඔබට අසාර්ථක සොයා ගැනීමේදී බොහෝ කාලයක් ගත කිරීමට හැකියාව ඇත. ඔබට ගැටළුව NP-complete බව පෙන්විය හැකි නම්, ඔබට එහිදී කාර්යක්ෂම algorithm එකක් සංවර්ධනය කිරීමට කාලය වැය කළ හැකිය, එය හොඳ, නමුත් හොඳම හැකියාවක් නොවේ.

සංකේතමය උදාහරණයක් ලෙස, මධ්‍යස්ථානයක් ඇති බෙදාහැරීමේ සමාගමක් සලකන්න. සෑම දිනකම, එය මධ්‍යස්ථානයේදී සෑම බෙදාහැරීමේ ට්‍රක් එකක්ම පූරණය කරයි සහ එය බොහෝ ලිපිනයන්ට භාණ්ඩ බෙදාහැරීමට යවයි. දවස අවසන් වන විට, සෑම ට්‍රක් එකක්ම මධ්‍යස්ථානයට ආපසු පැමිණිය යුතුය, එය ඊළඟ දිනට පූරණය කිරීමට සූදානම් වේ. වියදම් අඩු කිරීමට, සමාගමට බෙදාහැරීම් නියමිත කාලය තෝරා ගැනීමට අවශ්‍ය වේ, එය සෑම ට්‍රක් එකක්ම ගමන් කළ දුර අඩු කරයි. මෙම ගැටළුව ප්‍රසිද්ධ “traveling-salesman problem” ලෙස හැඳින්වේ, සහ එය NP-complete වේ. එයට කාර්යක්ෂම algorithm එකක් නොමැත. නමුත් විශේෂිත උපකල්පන යටතේ, අපට කාර්යක්ෂම algorithm එකක් දැනුම් ඇත, එය කුඩාම හැකියාවට වඩා ඉතා දුරක් නොවේ. අධ්‍යයනය 35 මෙවැනි “approximation algorithms” පිළිබඳ සාකච්ඡා කරයි.

සමාන්‍යතාව

බොහෝ වසරක් තිස්සේ, අපට processor clock speeds එකක් සමාන්‍ය අනුපාතයකින් වැඩිවීමක් ලෙස ගණනය කළ හැකි විය. නමුත් භෞතික සීමා කාලය වැඩිවීමේදී මූලික බාධාවක් ලෙස පවතී: බලය ඝනත්වය clock speed එක සමඟ superlinearly වැඩිවීම නිසා, chips එකක් උණුසුම් වීමේ අවදානමක් ඇති කරයි. එබැවින්, තත්පරයකට වැඩි ගණනක් ගණනය කිරීමට, chips එකක් නිර්මාණය කරනු ලබන්නේ එකක් පමණක් නොව, බොහෝ “cores” එකක් අඩංගු වීමයි. මෙම multicore පරිගණකවලට එකම chip එකක බොහෝ අනුක්‍රමික පරිගණකවලට සමාන ලෙස සමාන කළ හැක; වෙනත් වචන වලින්, එය “parallel computer” එකක් ලෙස හැඳින්විය හැක. Multicore පරිගණක වලින් හොඳම කාර්යක්ෂමතාවය ලබා ගැනීමට, අපට සමාන්‍යතාවය මත පදනම් වූ algorithm නිර්මාණය කළ යුතුය. අධ්‍යයනය 27 “multithreaded” algorithms සඳහා ආකෘතියක් ඉදිරිපත් කරයි, එම algorithms බොහෝ cores වල ප්‍රයෝජනය ගනී. මෙම ආකෘතිය තත්ත්වමය අංශයෙන් වාසියක් ඇත, සහ එය කිහිපයක් සාර්ථක පරිගණක වැඩසටහන් වල පදනමක් වේ, එමගින් ශාස්ත්‍රාපති චෙස් වැඩසටහනක් ඇතුළුව.

1.2 Algorithms as a technology

Exercises

1.1-1

Give a real-world example that requires sorting or a real-world example that requires computing a convex hull.

1.1-2

Other than speed, what other measures of efficiency might one use in a real-world setting?

1.1-3

Select a data structure that you have seen previously, and discuss its strengths and limitations.

1.1-4

How are the shortest-path and traveling-salesman problems given above similar? How are they different?

1.1-5

Come up with a real-world problem in which only the best solution will do. Then come up with one in which a solution that is “approximately” the best is good enough.

Algorithms as a technology

කම්පියුටර් ඉතා වේගවත් හා කම්පියුටර් මතකය නොමිලේ නම්, ඔබට algorithms අධ්‍යයනය කිරීමට කිසිදු හේතුවක් තිබේද? එහි පිළිතුර නම් ඔව්, ඔබේ විසඳුම් ක්‍රමය නිමාවට පත් වන බව සහ නිවැරදි පිළිතුරක් ලබා දෙන බව පෙන්වන්න කැමති නම්.

කම්පියුටර් ඉතා වේගවත් නම්, ගැටළුවක් විසඳීමට නිවැරදි ක්‍රමයක් භාවිතා කළ හැක. ඔබේ ක්‍රියාත්මක කිරීම හොඳ මෘදුකාංග ඉංජිනේරු ක්‍රියාවලියක් තුළ තිබිය යුතුය (උදාහරණයක් ලෙස, ඔබේ ක්‍රියාත්මක කිරීම හොඳින් නිර්මාණය කර ඇති සහ ලේඛනය කර ඇති බව), නමුත් ඔබට බොහෝ විට ක්‍රියාත්මක කිරීමට පහසුම ක්‍රමය භාවිතා කිරීමට කැමති විය හැක.

ඇත්ත වශයෙන්ම, කම්පියුටර් වේගවත් විය හැක, නමුත් ඒවා අසීමිත වේගවත් නොවේ. සහ මතකය අඩු මිලකට ලබා ගත හැක, නමුත් එය නොමිලේ නොවේ. එබැවින්, ගණනය කිරීමේ කාලය සීමාකාරී සම්පත් වේ, සහ මතකයේ ස්ථානයද එසේය. ඔබට මෙම සම්පත් ප්‍රයෝජනවත් ලෙස භාවිතා කළ යුතුය, සහ කාලය හෝ ස්ථානයේදී කාර්යක්ෂම වන algorithms ඔබට එය කිරීමට උපකාරී වේ.

12 කොටස 1 ගණිතමය ක්‍රමවේදයන්ගේ භූමිකාව

කාර්යක්ෂමතාව

එම ගැටලුව විසඳීමට නිර්මාණය කරන ලද විවිධ ගණිතමය ක්‍රමවේදයන්ගේ කාර්යක්ෂමතාවය බොහෝ විට දැඩි ලෙස වෙනස් වේ. මෙම වෙනස්කම් යන්ත්‍ර හා මෘදුකාංගය මගින් ඇති වෙනස්කම්ට වඩා වඩාත් වැදගත් විය හැක.

උදාහරණයක් ලෙස, කොටස 2 හි අපි වර්ගීකරණය සඳහා ගණිතමය ක්‍රමවේද දෙකක් බලන්නෙමු. පළමුව, "insertion sort" ලෙස හැඳින්වෙන ක්‍රමවේදය n අයිතම වර්ගීකරණය කිරීමට c_1n^2 කාලයක් ගනී, එහි c_1 යනු n මත රඳා නොපවතින ස්ථිරයක් වේ. එනම්, එය n^2 වලට සමාන කාලයක් ගනී. දෙවන ක්‍රමවේදය, "merge sort", $c_2n \lg n$ කාලයක් ගනී, එහි $\lg n$ යනු $\log_2 n$ වේ සහ c_2 යනු n මත රඳා නොපවතින තවත් ස්ථිරයක් වේ. "Insertion sort" සාමාන්‍යයෙන් "merge sort" ට වඩා කුඩා ස්ථිර සාධකයක් ඇත, එබැවින් $c_1 < c_2$ වේ.

අපි බලන්නෙමු කෙසේද ස්ථිර සාධක කාලය මත බලපෑමක් ඇති කරන්නේ ය. "Insertion sort" හි ක්‍රියාකාලය c_1n^2 ලෙස ලියමු සහ "merge sort" හි ක්‍රියාකාලය $c_2n \lg n$ ලෙස ලියමු. එවිට අපි දැක්කොත් "insertion sort" හි ක්‍රියාකාලයේ n සාධකයක් ඇත, "merge sort" හි $\lg n$ සාධකයක් ඇත, එය වඩාත් කුඩා වේ. (උදාහරණයක් ලෙස, n = 1000 වන විට, $\lg n$ සමානව 10 වේ, සහ n = 1,000,000 වන විට $\lg n$ සමානව 20 ක් පමණ වේ.) "Insertion sort" සාමාන්‍යයෙන් කුඩා ආදාන ප්‍රමාණ සඳහා "merge sort" ට වඩා වේගවත් ක්‍රියාත්මක වේ, නමුත් ආදාන ප්‍රමාණය n ප්‍රමාණවත් විශාල විය හැකි විට, "merge sort" හි $\lg n$ සහ n අතර වාසිය ස්ථිර සාධක වෙනස්කම් සඳහා වඩාත් ප්‍රමාණවත් වේ. c_1 c_2 ට වඩා කුඩා වුවද, "merge sort" වේගවත් වන සීමාවක් සෑම විටම තිබේ.

සත්‍ය උදාහරණයක් ලෙස, අපි "insertion sort" ක්‍රමවේදය ක්‍රියාත්මක කරන වේගවත් පරිගණකයක් (පරිගණක A) සහ "merge sort" ක්‍රමවේදය ක්‍රියාත්මක කරන මන්දගාමී පරිගණකයක් (පරිගණක B) අතර තරඟයක් පැවැත්වීම සිතා බලමු. ඔවුන්ට 10 මිලියන සංඛ්‍යාවන්ගේ අරය වර්ගීකරණය කළ යුතුය. (10 මිලියන සංඛ්‍යාවන් බොහෝ බවක් පෙනෙන්නට තිබුණත්, එම සංඛ්‍යාවන් අට බයිට් පූර්ණ සංඛ්‍යාවන් නම්, එම ආදානය සම්පූර්ණයෙන්ම 80 මැගාබයිට් පමණ වේ, එය සාමාන්‍ය laptop පරිගණකයක මතකය තුළ ගැළපේ.) පරිගණක A එක තත්පරයකදී 10 බිලියන උපදෙස් ක්‍රියාත්මක කරන බව සලකන්න (මෙම ලිපිය ලියන විට ඕනෑම එක් අනුපිළිවෙලක පරිගණකයකට වඩා වේගවත්) සහ පරිගණක B එක තත්පරයකදී 10 මිලියන උපදෙස් ක්‍රියාත්මක කරන බව සලකන්න, එබැවින් පරිගණක A එක පරිගණක B එකට වඩා කුඩා ගණනක් වේ. මෙම වෙනස තවත් දැඩි කිරීමට, ලෝකයේ හොඳම වැඩසටහනකරු පරිගණක A සඳහා "insertion sort" යන්ත්‍ර භාෂාවෙන් කේතය ලිවීම සිතා බලමු, සහ එම කේතය n සංඛ්‍යාවන් වර්ගීකරණය කිරීමට $2n^2$ උපදෙස් අවශ්‍ය වේ. තවද, සාමාන්‍ය වැඩසටහනකරු "merge sort" ක්‍රමවේදය ඉහළ මට්ටමේ භාෂාවකින් ක්‍රියාත්මක කරන බව සලකන්න, අසාර්ථක සම්පීඩකයක් භාවිතා කරමින්, එම කේතය $50n \lg n$ උපදෙස් ගනී. 10 මිලියන සංඛ්‍යාවන් වර්ගීකරණය කිරීමට, පරිගණක A එකට 2 .107/2 උපදෙස් අවශ්‍ය වේ, එය 20,000 තත්පර (පහළම 5.5 පැය) වේ; 10^{10} උපදෙස්/තත්පර D, එ enquanto පරිගණක B එකට...

1.2 Algorithms as a technology

50 107 lg 107 instructions " 1163 seconds (less than 20 minutes) : 107 instructions/second # ඉතා අඩු කාලයක් තුළ ක්‍රියාත්මක වන algorithm එකක් භාවිතා කිරීමෙන්, කුඩා compiler එකක් තිබුණද, පරිගණක B, පරිගණක A ට වඩා වඩාත් වේගවත් ක්‍රියාත්මක වේ! merge sort එකේ වාසිය, 100 මිලියන සංඛ්‍යා සංඛ්‍යාත කිරීමට අවශ්‍ය වූ විට, ඉතා පැහැදිලි වේ: insertion sort එකට 23 දිනකට වඩා වැඩි කාලයක් ගත වන අතර, merge sort එකට පැය 4 කින් අඩු වේ. සාමාන්‍යයෙන්, ගැටලුවේ ප්‍රමාණය වැඩි වීමත් සමඟ, merge sort එකේ සRelative advantage එකද වැඩි වේ.

Algorithms and other technologies

ඉහත උදාහරණය පෙන්වා දෙන්නේ, අපි algorithms, පරිගණක hardware එකක් වැනි, තාක්ෂණයක් ලෙස සලකා බැලිය යුතුය. සම්පූර්ණ පද්ධතියේ කාර්ය සාධනය, වේගවත් hardware එකක් තෝරා ගැනීම මෙන්ම කාර්යක්ෂම algorithms තෝරා ගැනීම මතද පදනම් වේ. අනෙක් පරිගණක තාක්ෂණයන්හි වේගවත් ප්‍රගතියක් සිදු වන්නේ පරිගණක algorithms වලද වේ.

ඔබට අසන්න පුළුවන්, contemporary පරිගණකවල algorithms එතරම් වැදගත්ද යන්න, අනෙකුත් ප්‍රවීණ තාක්ෂණයන්, වැනි:

- advanced computer architectures සහ fabrication technologies,
- easy-to-use, intuitive, graphical user interfaces (GUIs),
- object-oriented systems,
- integrated Web technologies, සහ
- fast networking, both wired and wireless.

ආපසු පිළිතුර වන්නේ ඔව්. කිහිපයක්ම applications වල algorithmic content අවශ්‍ය නොවේ (උදාහරණයක් ලෙස, කිහිපයක්ම සරල Web-based applications), නමුත් බොහෝ applications වල අවශ්‍ය වේ. උදාහරණයක් ලෙස, එක් Web-based සේවාවක් සලකා බලන්න, එය එක් ස්ථානයකින් අනෙක් ස්ථානයට ගමන් කිරීමට කෙසේද යන්න තීරණය කරයි. එහි ක්‍රියාත්මක කිරීම වේගවත් hardware එකක්, graphical user interface එකක්, wide-area networking එකක්, සහ object orientation එකක් මත පදනම් වේ. නමුත්, එය routes සොයා ගැනීම (probably using a shortest-path algorithm), maps rendering කිරීම, සහ addresses interpolate කිරීම වැනි විශේෂිත ක්‍රියාකාරකම් සඳහා algorithms අවශ්‍ය වේ.

ඉතිරියට, algorithmic content අවශ්‍ය නොවන application එකක්ද algorithms මත දැඩි ලෙස පදනම් වේ. එම application එක වේගවත් hardware එකක් මත පදනම් වේද? Hardware design එක algorithms භාවිතා කරයි. එම application එක graphical user interfaces මත පදනම් වේද? GUI එකක ඕනෑම design එක algorithms මත පදනම් වේ. එම application එක networking එකක මත පදනම් වේද? Networks වල routing algorithms මත දැඩි ලෙස පදනම් වේ. එම application එක machine code එකක් නොවන භාෂාවකින් ලියන ලද්දේද? එවිට එය compiler, interpreter, හෝ assembler එකක් මඟින් සැකසී ඇත, එම සියල්ලම algorithms භාවිතා කරයි.

1 වන කොටස: ගණිතමය ක්‍රමවේද වල භූමිකාව

ගණිතමය ක්‍රමවේද යනු නවීන පරිගණක වල භාවිතා කරන බොහෝ තාක්ෂණයන්ගේ මූලික කොටස වේ.

ඉතා වැඩි පරිගණක හැකියාවන් සමඟ, අපි එම පරිගණක භාවිතා කරමින් කලින් කිසිදා නොවූ විශාල ගැටළු විසඳීමට යොමු වෙමු. ඉහත සඳහන් වූ insertion sort සහ merge sort අතර සසඳන විට, ගණිතමය ක්‍රමවේද අතර කාර්යක්ෂමතාවයේ වෙනස්කම් විශේෂයෙන් ප්‍රධාන වේ.

ගණිතමය දැනුම සහ තාක්ෂණය පිළිබඳ ශක්තිමත් පදනමක් තිබීම, සැබෑවටම කුසලතා ඇති වැඩසටහන් නිර්මාතෘන් සහ අලුත්කාරයන් අතර වෙනසක් වේ. නවීන පරිගණක තාක්ෂණය සමඟ, ඔබට ගණිතමය ක්‍රමවේද පිළිබඳ විශාල දැනුමක් නැතිවත් කිහිපයක් කළ හැක, නමුත් ගණිතමය ක්‍රමවේද පිළිබඳ හොඳ පසුබැසීමක් ඇතිව, ඔබට ඉතා, ඉතා වැඩි දේ කළ හැක.

අභියෝග

1.2-1

ගණිතමය අන්තර්ගතයක් අවශ්‍ය කරන යෙදුමක් උදාහරණයක් දෙන්න, සහ එහි සම්බන්ධ ගණිතමය ක්‍රමවේද වල කාර්යය සාකච්ඡා කරන්න.

1.2-2

අපි එකම යන්ත්‍රයේ insertion sort සහ merge sort ක්‍රමවේදයන්ගේ ක්‍රියාත්මක කිරීම සසඳනවා යැයි හිතමු. n ප්‍රමාණයේ ආදාන සඳහා, insertion sort ක්‍රමවේදය $8n^2$ පියවර වලින් ක්‍රියාත්මක වේ, එසේම merge sort ක්‍රමවේදය $64n \lg n$ පියවර වලින් ක්‍රියාත්මක වේ. insertion sort ක්‍රමවේදය merge sort ක්‍රමවේදය වඩා කුමන n අගයන්හි ජයග්‍රහණය කරයිද?

1.2-3

n හි කුඩාම අගය කුමක්ද, එහි ක්‍රියාත්මක වේලාව $100n^2$ වන ගණිතමය ක්‍රමවේදයක්, එකම යන්ත්‍රයේ ක්‍රියාත්මක වේලාව $2n$ වන ගණිතමය ක්‍රමවේදයක් වඩා වේගවත් ක්‍රියාත්මක වේද?

ගැටළු

1-1: ක්‍රියාත්මක වේලාවන්ගේ සසඳීම

ඉදිරිපත් කරන ලද මේ වගුවේ සෑම ක්‍රියාවලියක් $f(n)$ සහ වේලාව t සඳහා, ගැටළුවක් විසඳීමට t වේලාව තුළ විසඳිය හැකි ප්‍රශ්නයේ විශාලතම n ප්‍රමාණය තීරණය කරන්න, ගැටළුව විසඳීමට $f(n)$ මයික්‍රොසෙක්න්ඩ් ගත වන බව හිතන විට.

Chapter 1 සඳහා සටහන්

```
              1       1       1       1       1       1       1
              second   minute    hour     day    month     year    century
        lg n
pn


   n
  n lg n
     n2
     n3
      2n

       nŠ
```

Chapter notes

අල්ගොරිතම් පිළිබඳ සාමාන්‍ය විෂයය සඳහා විශිෂ්ට පෙළ ගණනාවක් ඇත, එමෙන්ම Aho, Hopcroft, සහ Ullman [5, 6]; Baase සහ Van Gelder [28]; Brassard සහ Bratley [54]; Dasgupta, Papadimitriou, සහ Vazirani [82]; Goodrich සහ Tamassia [148]; Hofri [175]; Horowitz, Sahni, සහ Rajasekaran [181]; Johnsonbaugh සහ Schaefer [193]; Kingston [205]; Kleinberg සහ Tardos [208]; Knuth [209, 210, 211]; Kozen [220]; Levitin [235]; Manber [242]; Mehlhorn [249, 250, 251]; Purdom සහ Brown [287]; Reingold, Nievergelt, සහ Deo [293]; Sedgewick [306]; Sedgewick සහ Flajolet [307]; Skiena [318]; සහ Wilf [356] විසින් ලියන ලද කෘති. අල්ගොරිතම් නිර්මාණයේ වඩාත් ප්‍රායෝගික පැතිකඩ කිහිපයක් Bentley [42, 43] සහ Gonnet [145] විසින් සාකච්ඡා කර ඇත. අල්ගොරිතම් ක්ෂේත්‍රයේ සමාලෝචන හඳුනාගැනීමට හැකි වේ Theoretical Computer Science, Volume A [342] සහ CRC Algorithms and Theory of Computation Handbook [25] හි. ගණනාවෙන් පරිගණක ජීව විද්‍යාවට භාවිතා කරන අල්ගොරිතම් පිළිබඳ සාරාංශයක් Gusfield [156], Pevzner [275], Setubal සහ Meidanis [310], සහ Waterman [350] විසින් ලියන ලද පෙළ ගණනාවක සොයාගත හැක.